Cloud Robotics Azure Platform V1
Hands-on tranning session

Construction of basic set ver.1.4

Seiji Miyahara
Cloud Solutions Architect – Cloud AI, Analytics & Big Data

Microsoft Japan Co., Ltd.
Azure Solution Technology dept.
Application & Infrastructure Special Team Unit

Terms of use

Cloud Robotics Azure Platform V1 SDK (Software Development Kit) including tools, Cloud Robotics FX (Application Framework) and related software used in this hands-on session are sample code. These are provided as source code.

This sample code have to be used under the responsibility of the user and not to be held liable for defect warranty at our company.

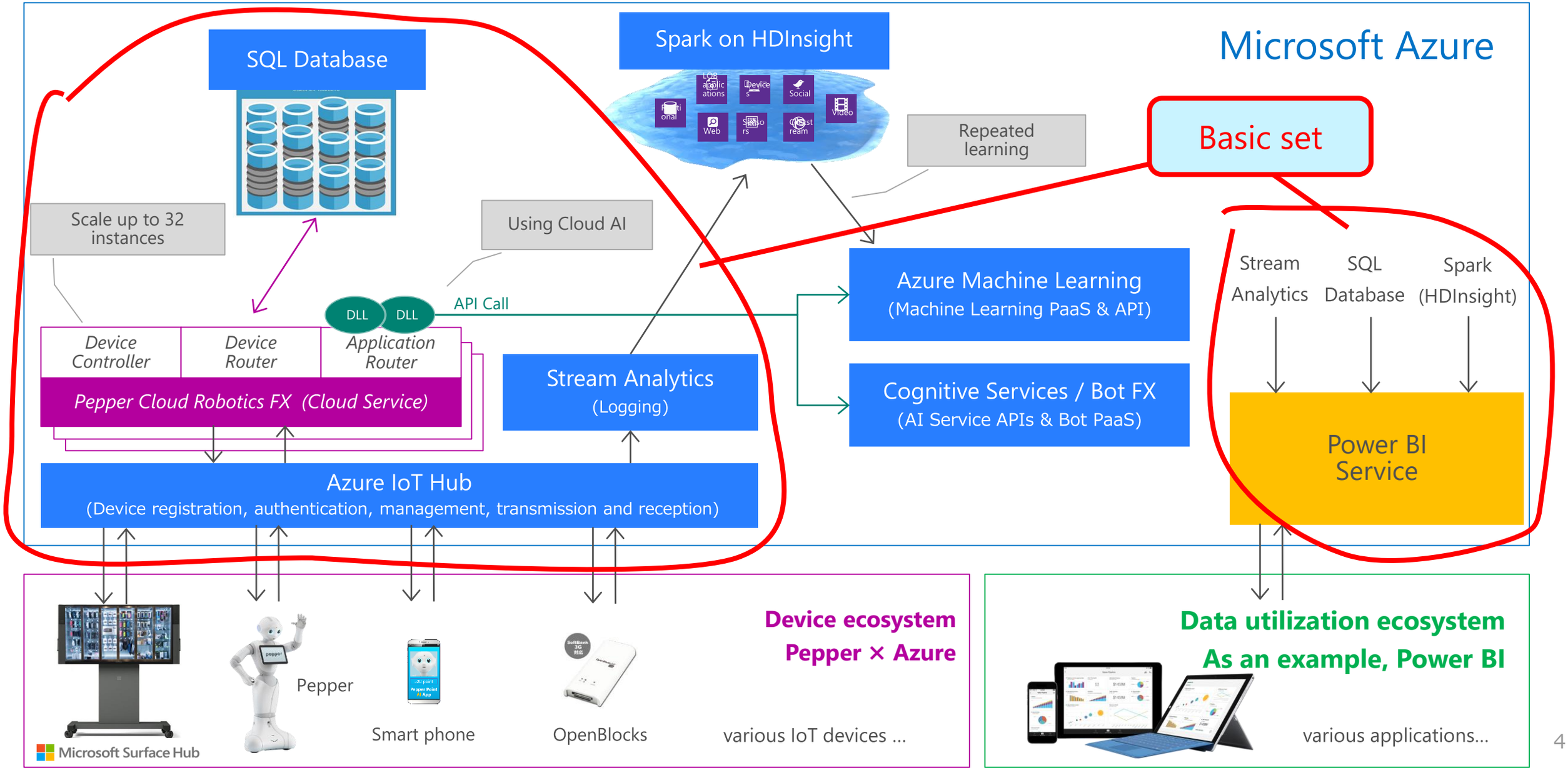
Preparation-1

1. Please prepare Azure Subscription.
2. Please check login to Azure Portal (portal.azure.com) and check privilege (Can you create a virtual machine etc.)
3. Please bring Windows PC (Windows 7 or later, recommended, Windows 10)
4. Please install Microsoft Azure Storage Explorer
   http://storageexplorer.com/
5. Please install SQL Server 2016 Management Studio
   https://msdn.microsoft.com/en-us/library/mt238290.aspx
6. Please install Visual Studio 2015 Pro or higher. If you do not have the license, please install the Community version.
   https://www.microsoft.com/en-us/dev/products/community.aspx
7. Please install Azure SDK for Visual Studio 2015
   https://go.microsoft.com/fwlink/?LinkId=518003&clcid=0x411
8. Please sign up & sign in for Power BI service
   http://powerbi.com/
   ※ Please register with the company's e-mail address (organization account only valid. Microsoft account or free-mail address can not be used)

Cloud Robotics Azure Platform

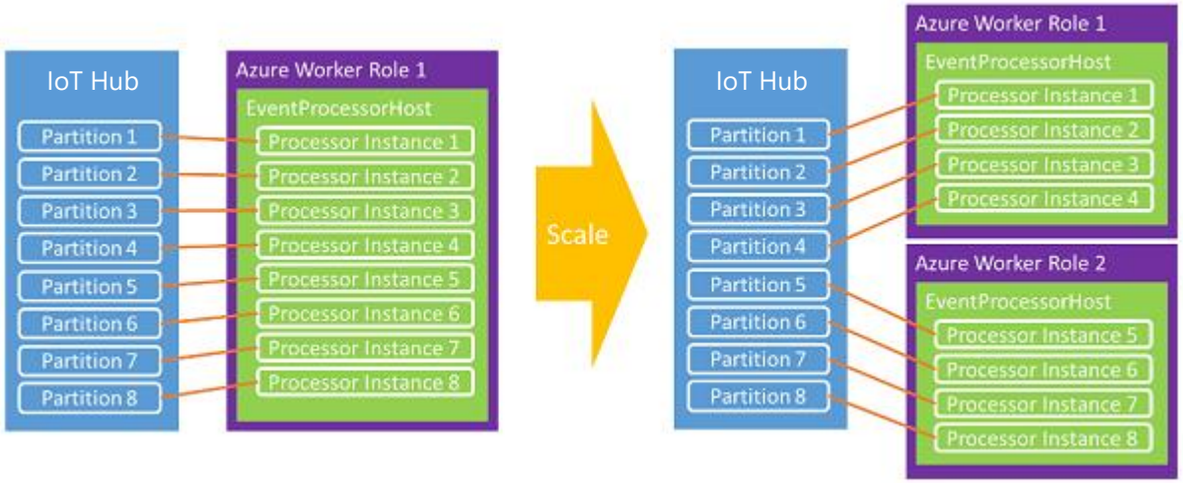
: Azure PaaS

: Robotics SDK



Cloud Robotics FX (App Framework Sample Code)

- Based on Event Processor Host Class
  - Scalable event processing framework
  - https://msdn.microsoft.com/ja-jp/library/azure/microsoft.servicebus.messaging.eventprocessorhost(v=azure.95).aspx
- Device Controller
  - Transmit control information to devices
    - Initial processing immediately after device activation
    - Control application updating on device
- Device Router
  - Device to device routing
    - 1 : 1 communication
    - 1 : n communication (group)
  - Do nothing if there is no routing definition,



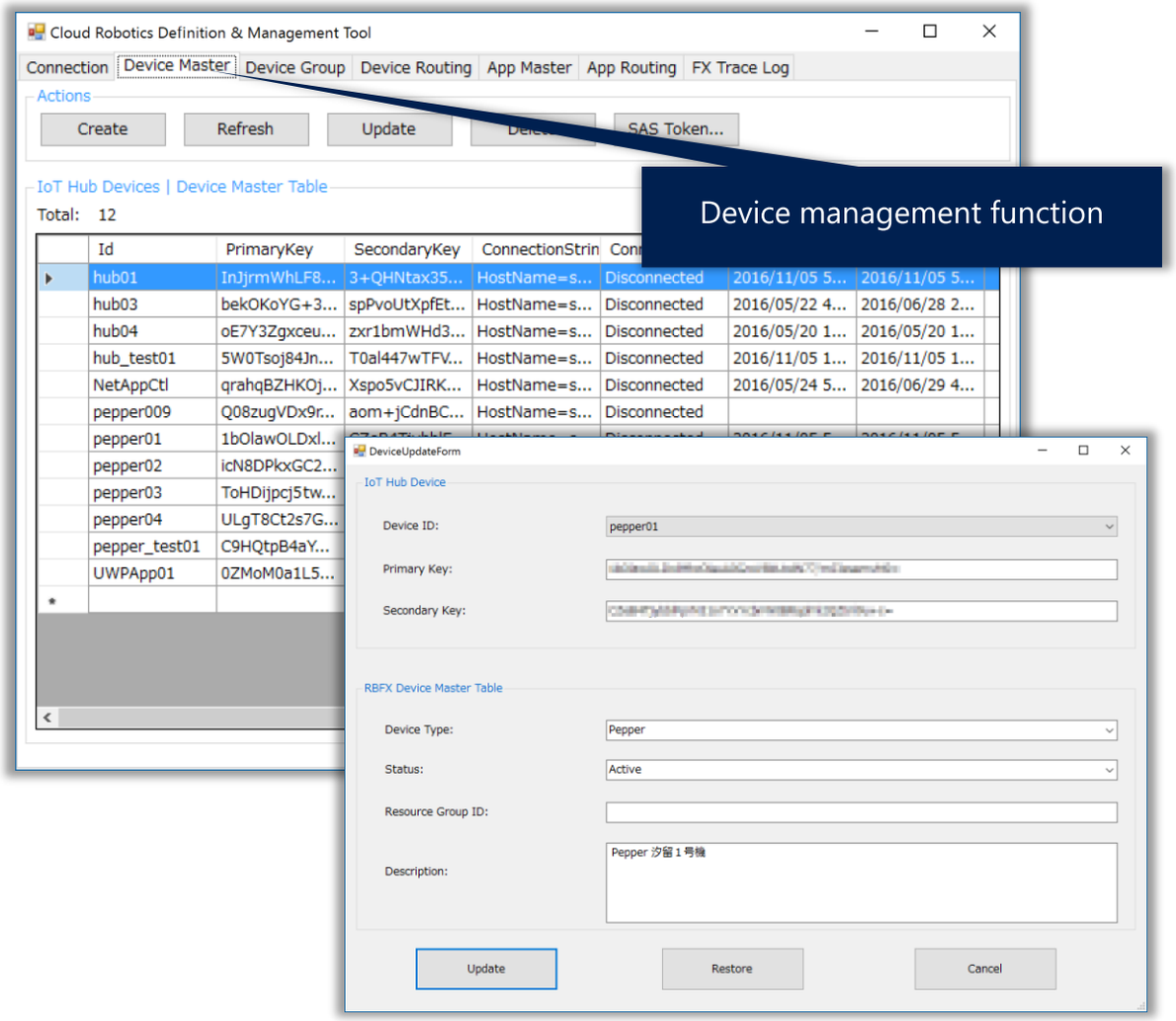
- Application Router
  - Dynamically read classes (DLL) which process received data from Device from BLOB
  - The destination of output results is specified from data processing class(DLL) or Device Router
  - A data processing class (DLL) for commonly used scenarios is prepared as a package

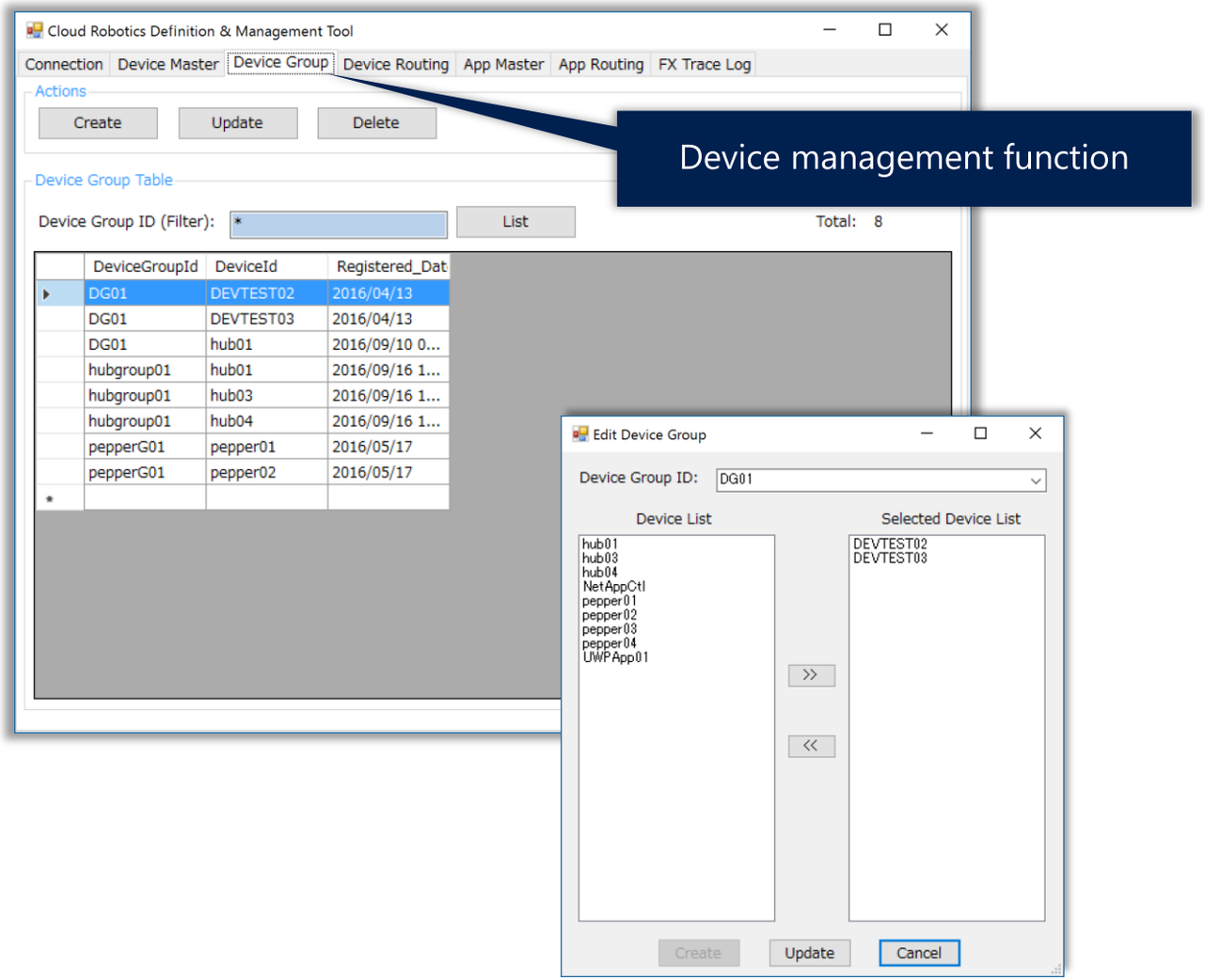
Cloud Robotics SDK

- Cloud Robotics FX Source code
- Sample code to learn the procedure with Cloud Robotics FX (IoT Hub)
  - C# (.NET)
  - Java
  - Python
- Cloud Robotics Definition & Management Tool
  - Register / Change / Delete / Check status and change / Check message content functions for device
  - Register / change / delete functions for Device Routing
  - Register / Change / Delete functions for Device Group
  - Register / Change / Delete / Check Status and Modify / Deploy DLL functions for device
  - Register / change / delete for Application Routing
  - Refer to Cloud Robotics FX trace log
- Cloud Robotics Device Simulator
  - Simulator for sending and receiving of Cloud Robotics FX compliant devices (including Pepper)
- Cloud Robotics Test Driver
  - Test driver for unit test of application (DLL) running on Cloud Robotics FX
- Standard Apps on Cloud Robotics FX
  - RbSampleApp : “Hello world !!” Application
  - RbCogFaceApi : A standard scenario-based Face API application

Cloud Robotics Definition Tool

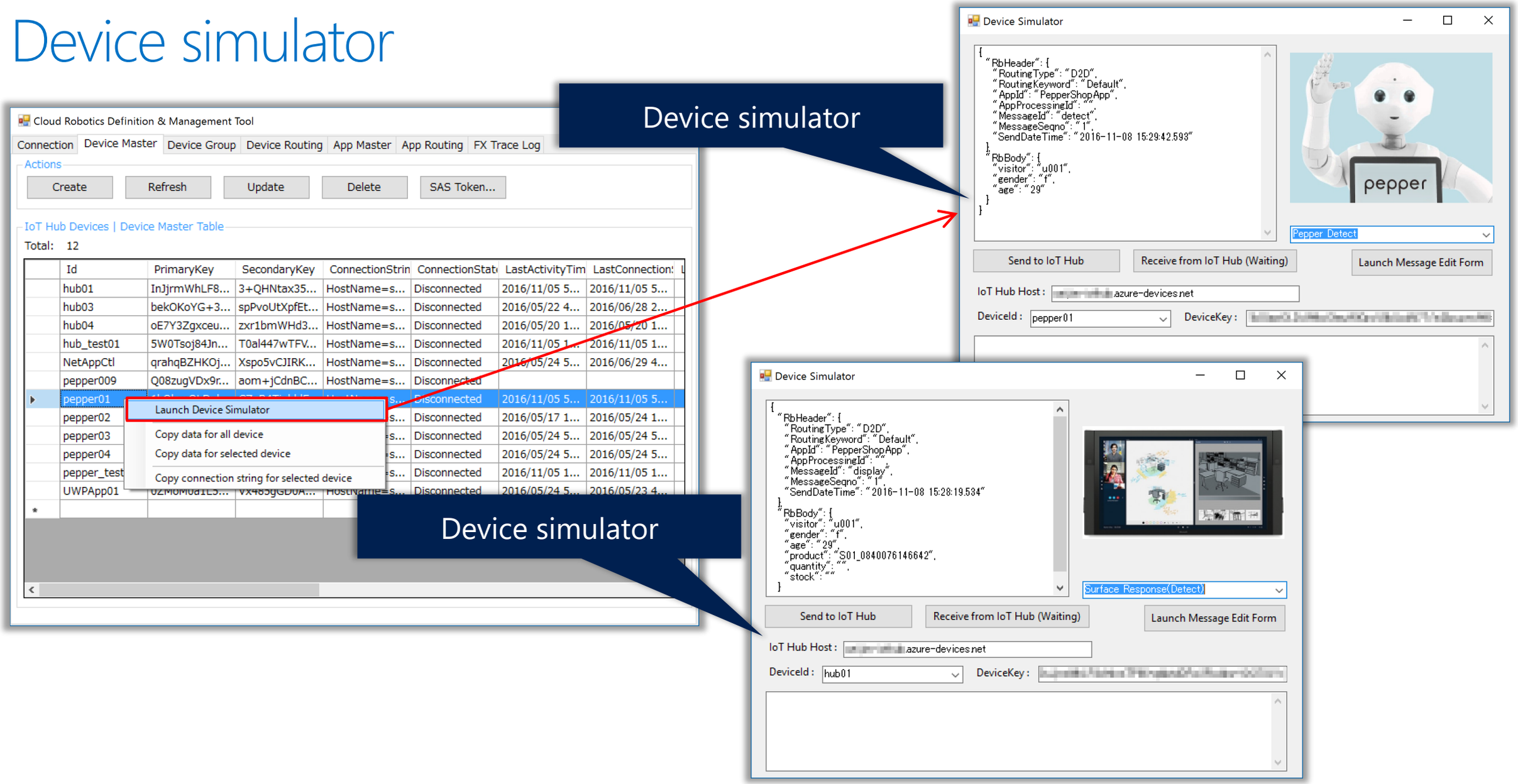
GUI Management Tools (Including IoT Hub Device Management functions)



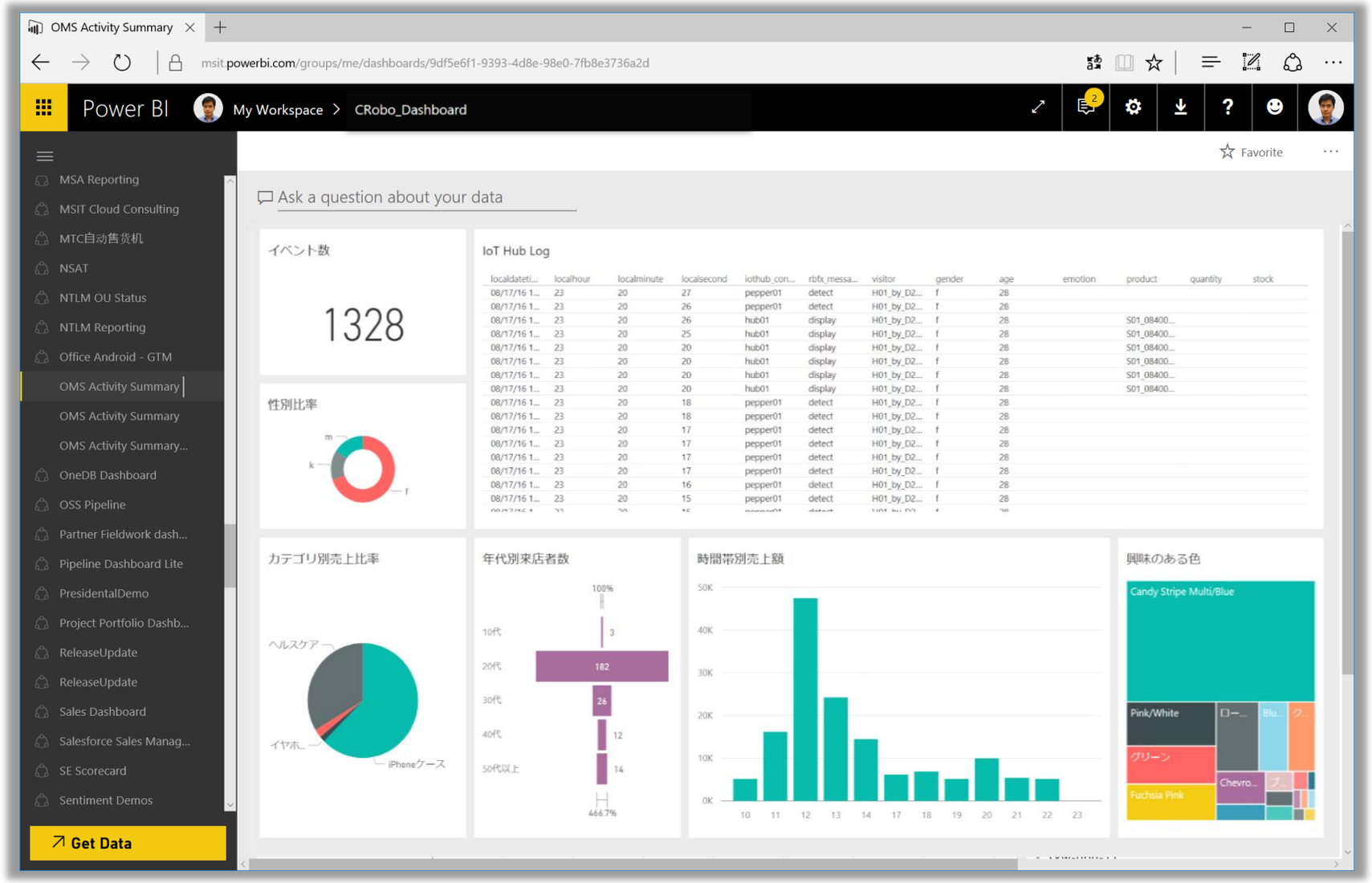


Cloud Robotics Device Simulator

Device simulator



Cloud Robotics Power BI Template



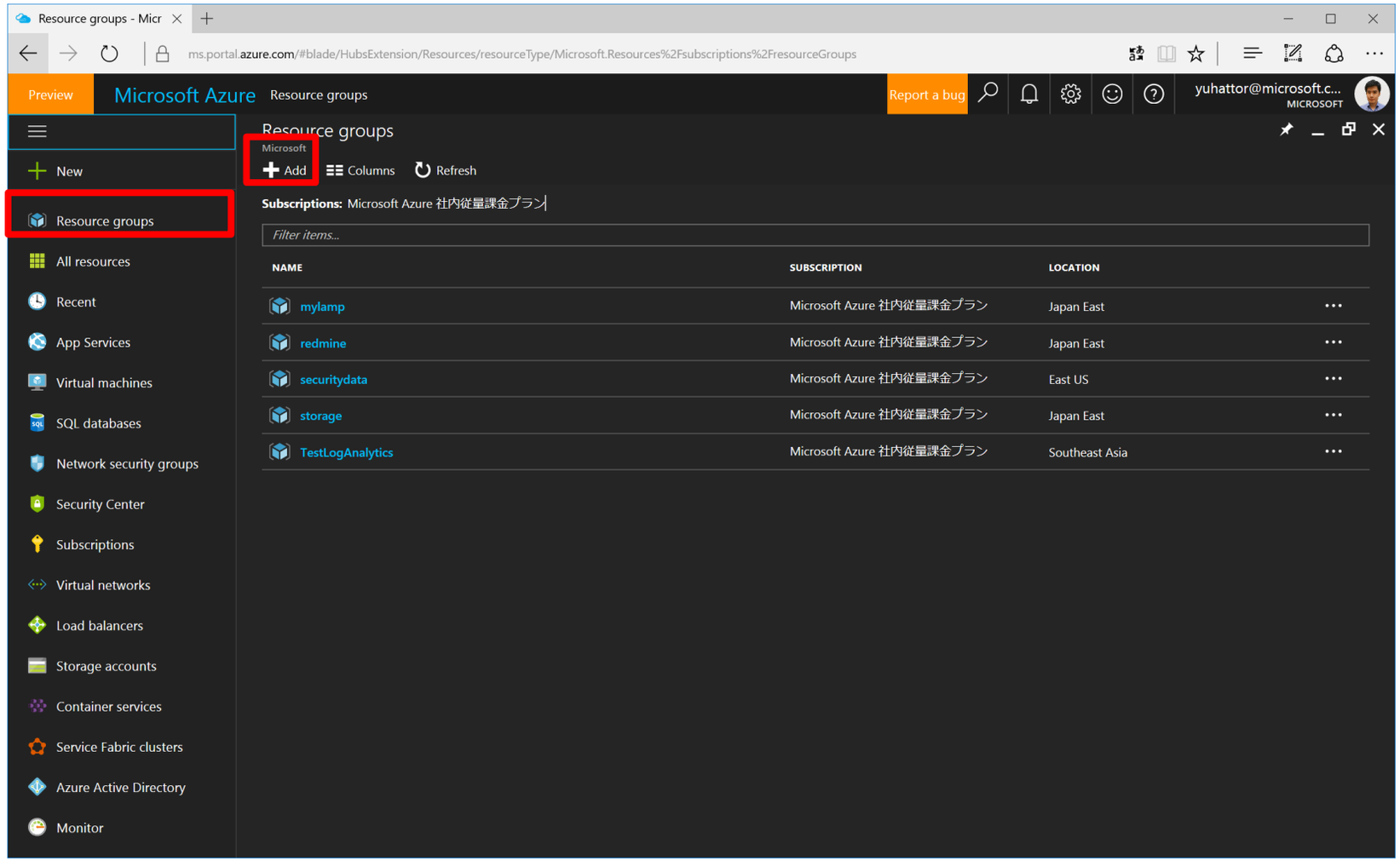
Tutorial 01

Create resource group storage

T01_01 : Create a new resource group

Log in to the Azure portal (portal.azure.com) and check whether it is the right subscription or not.

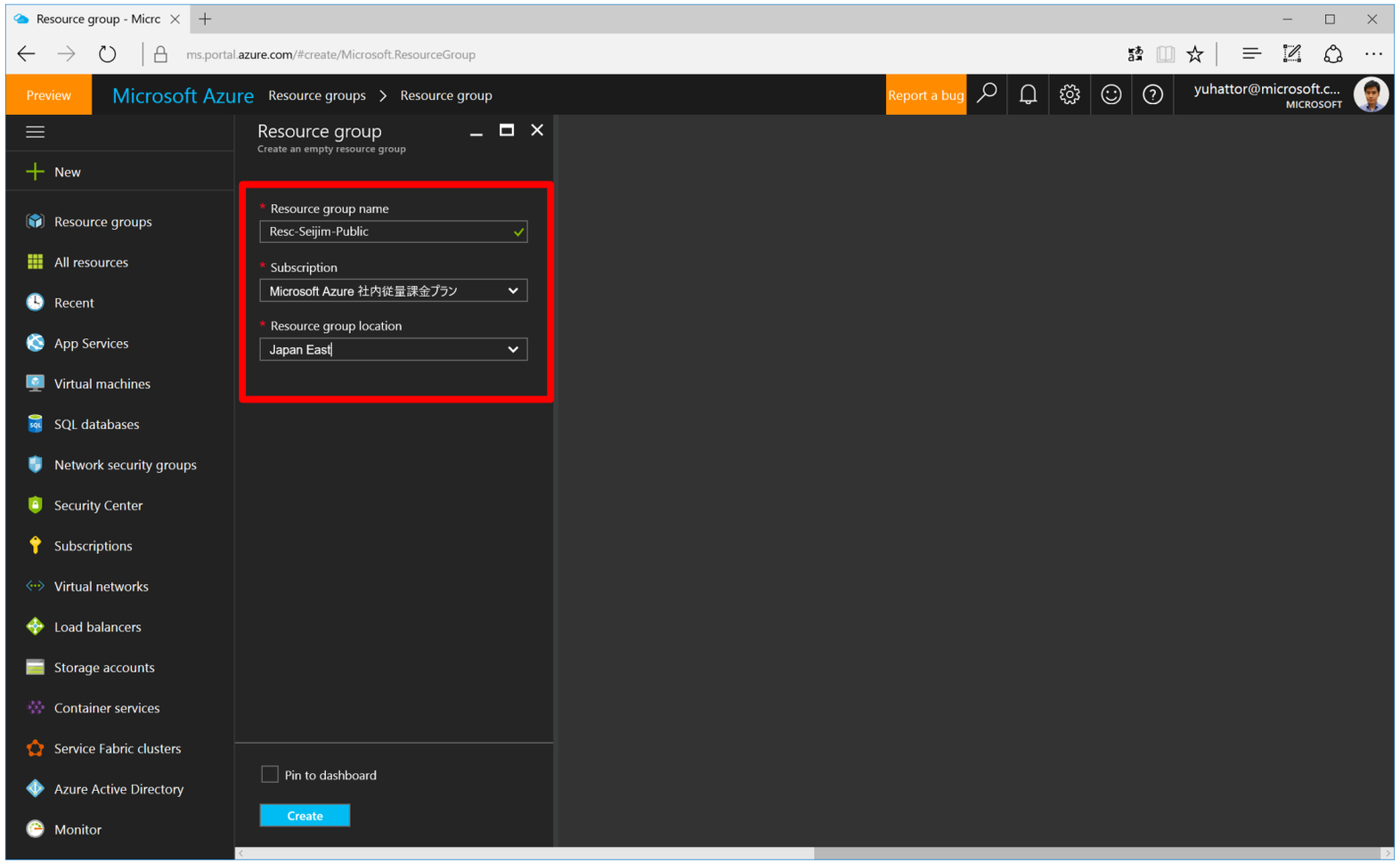
- [Resource Group] → [+Add]



T01_02 : Input Parameters for Resource Group

Set parameters as below and press the [Create] button

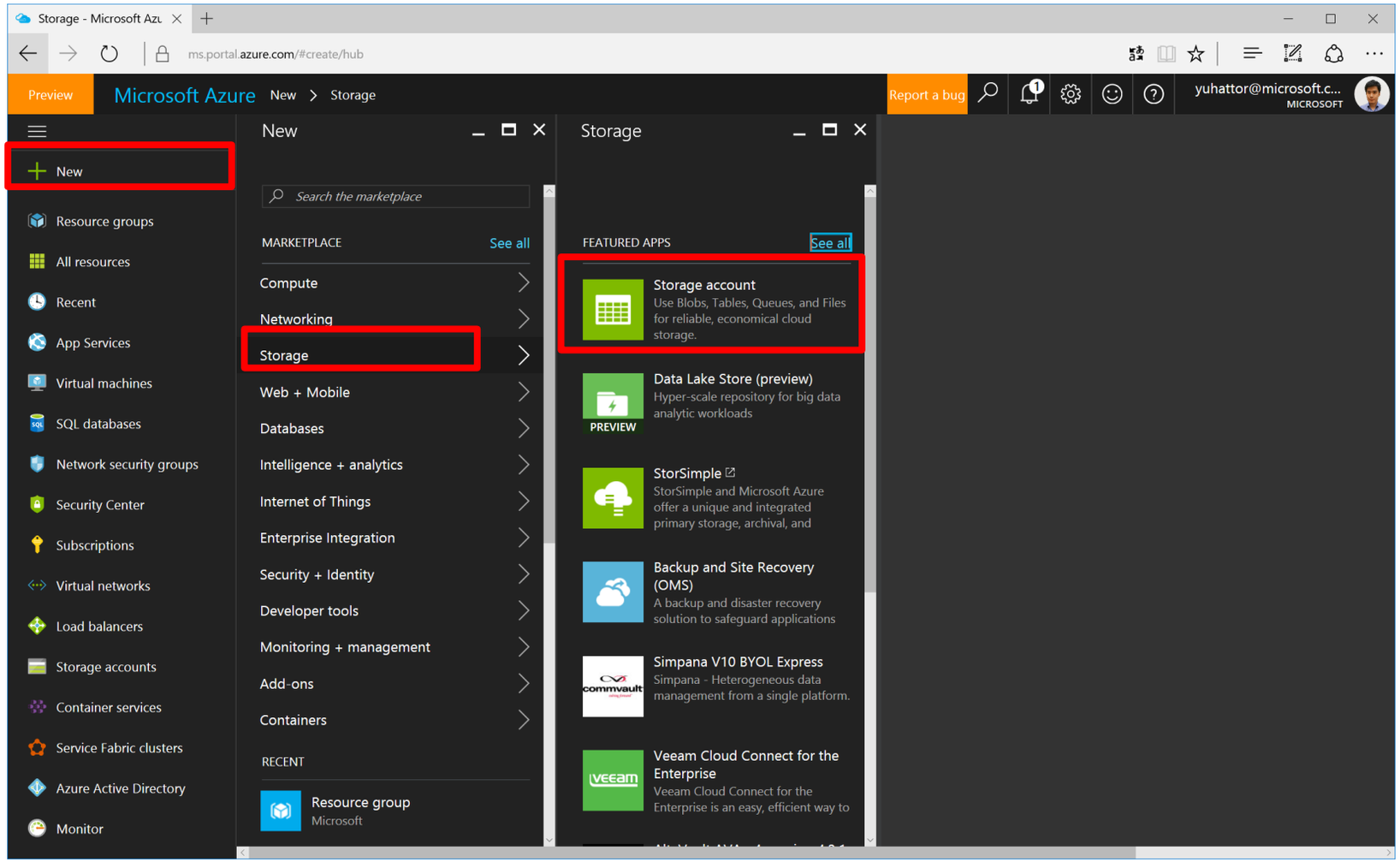
- [Resource group name] = {manageable name}
- [Resource group location] = {East Japan or West Japan}



T01_03 : Create a new storage account

Log in to the Azure portal (portal.azure.com) and check whether it is the right subscription or not.

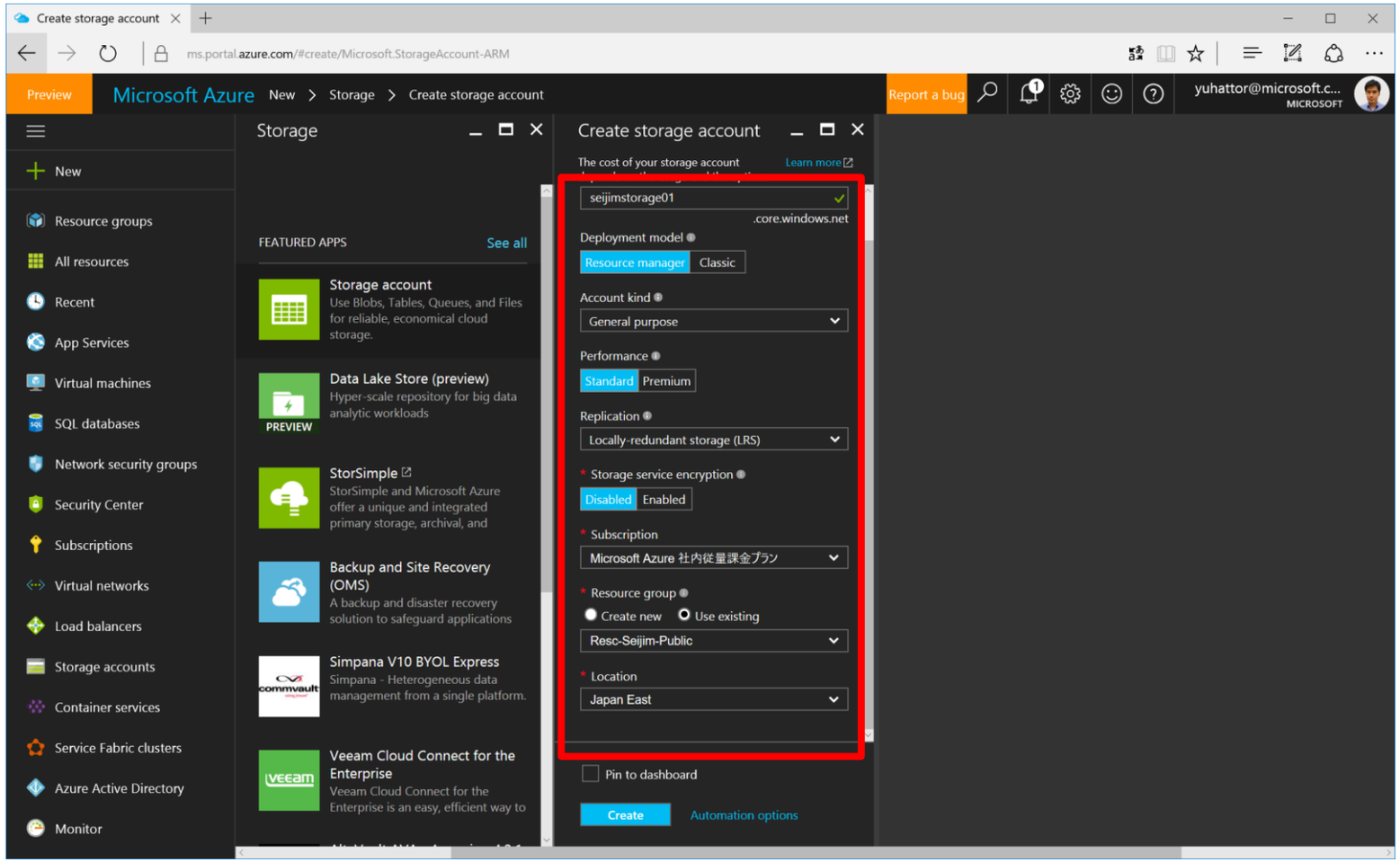
- [+ New] → [Storage] → [Storage account]



T01_04 ： Input Parameters for Storage account

Set the parameters as below and press the [Create] button

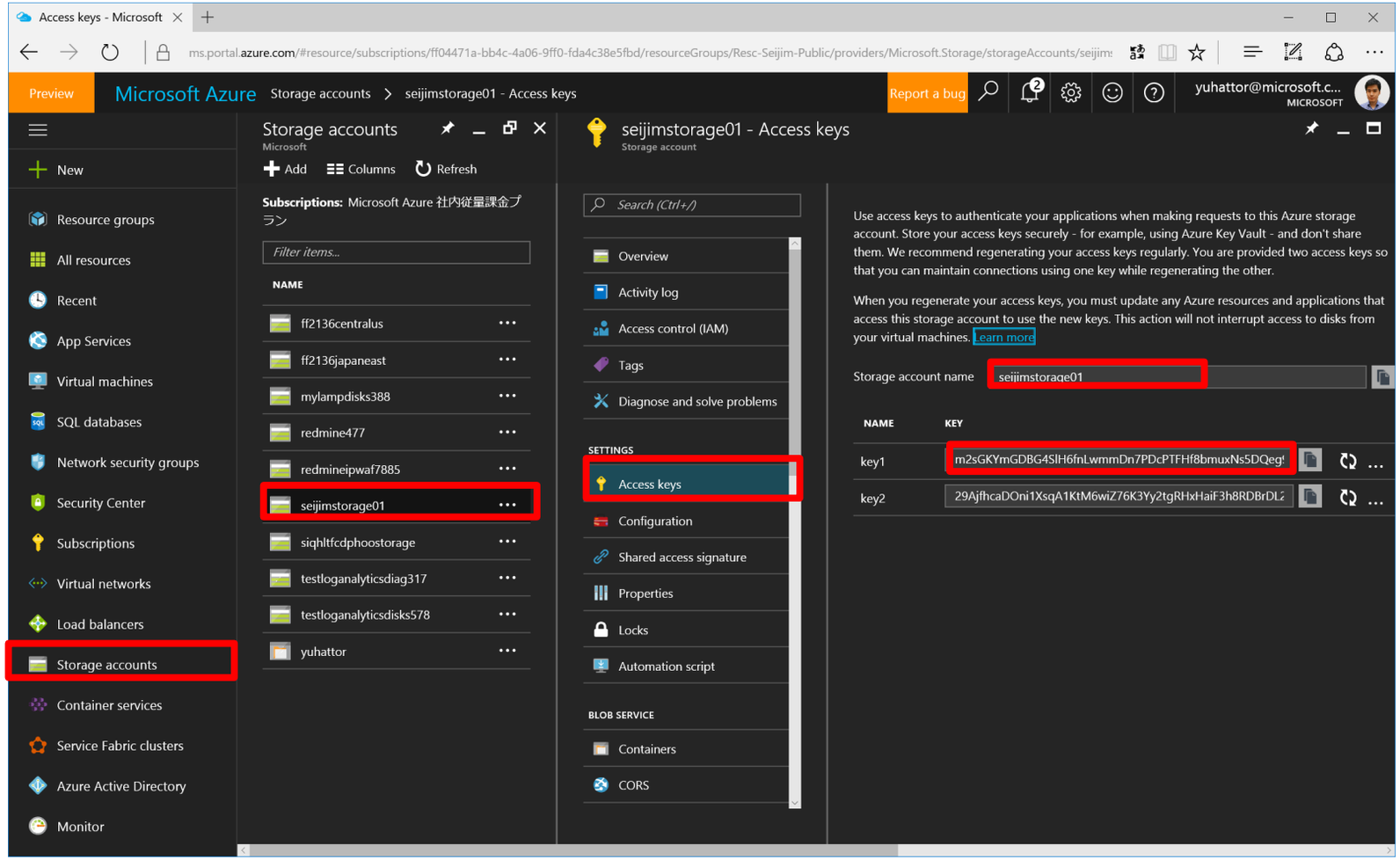
- [Name] = {manageable name}
- [Replication] = {Local Redundancy (LRS)}
- [Resource group] = {Set value of T01_02}
- [Location] = {East Japan or West Japan}



T01_05 : Confirm storage account

Select from the Azure portal as follows

- [Storage account] → [Applicable storage account] → [Access key]
- Save [Storage account name] and [key1] in Notepad etc.



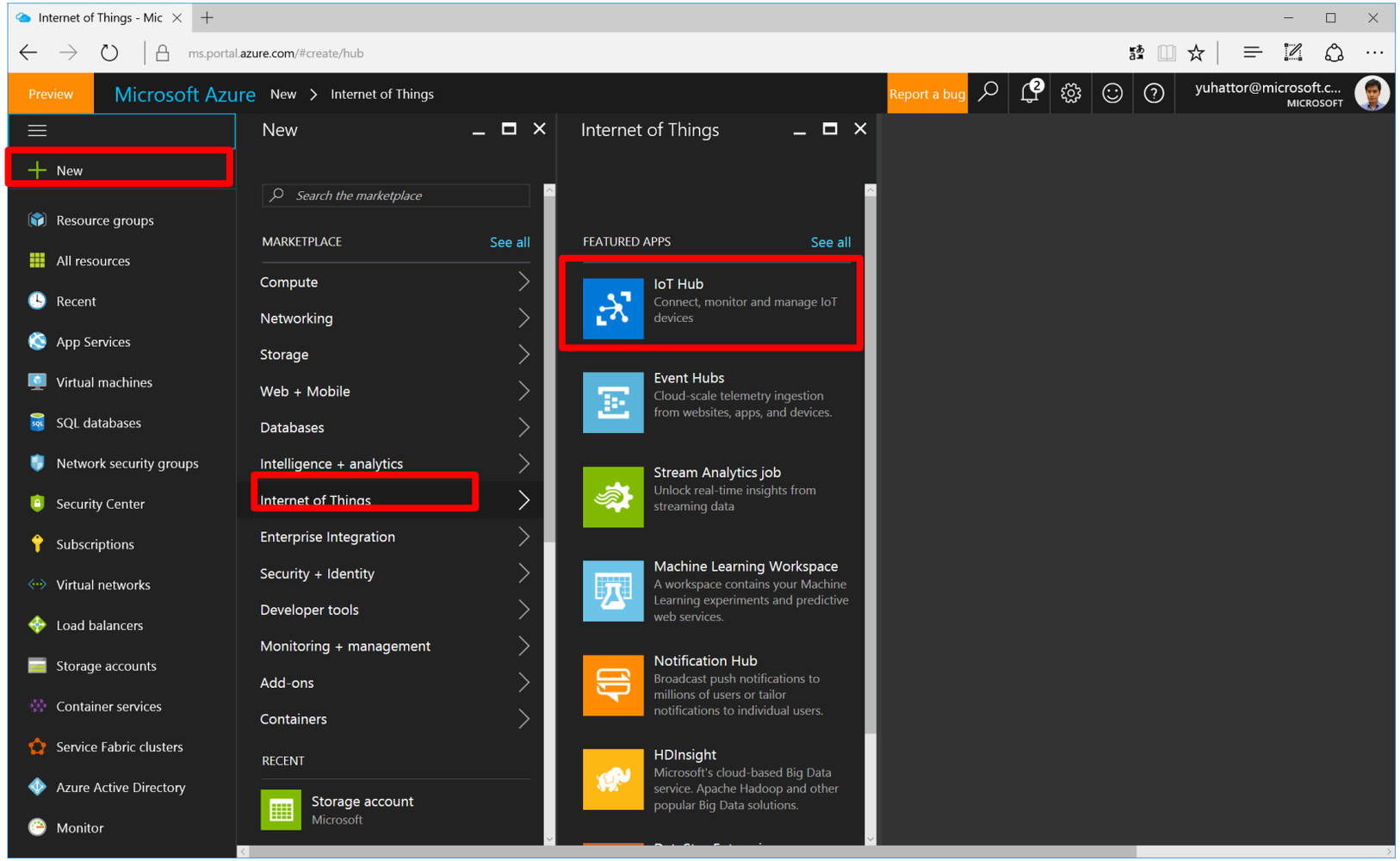
Tutorial 02

Create Azure IoT Hub

T02_01 : Create Azure IoT Hub

Select from the Azure portal as follows

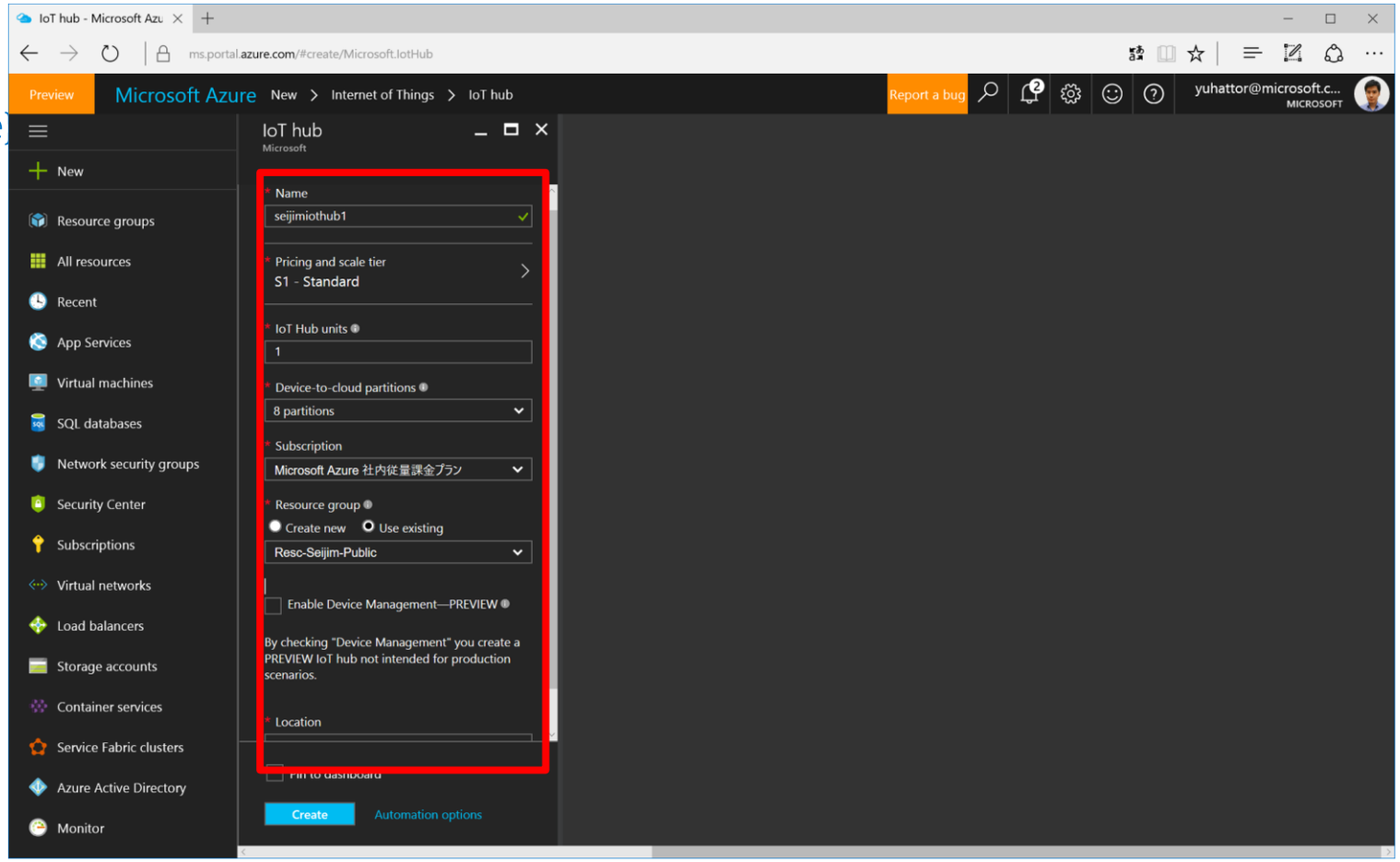
- [New] → [Internet of Things] → [IoT Hub]



T02_02 : Input parameters for IoT Hub

Set the parameters as below and press the [Create] button

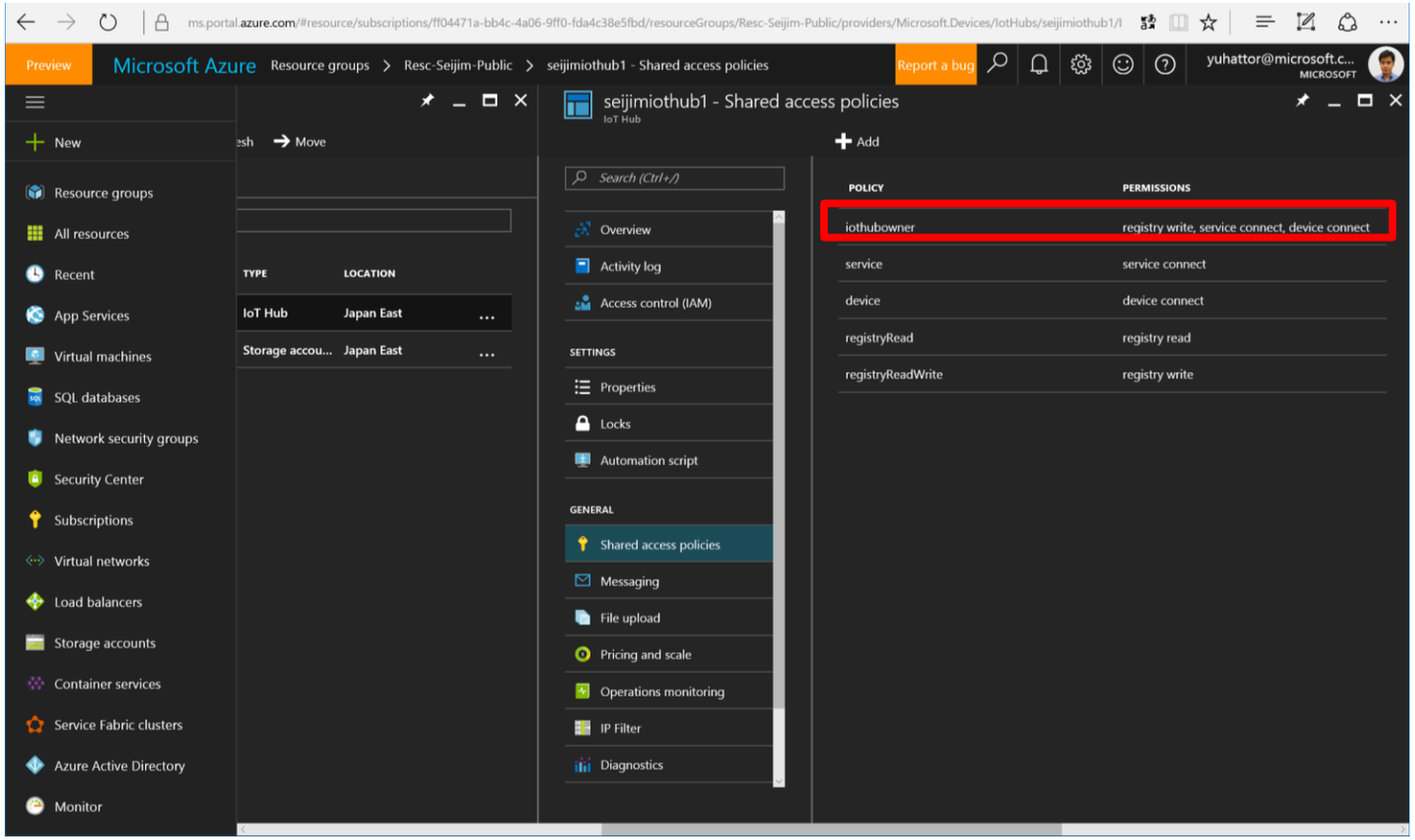
- [Name] = {Set lowercase letters without duplication}
- [Device-to-cloud partition] = {16}
- [Resource group] = {Set value of T 01}
- [Enable Device Management - Preview] = {None}
- [Location] = {East Japan or West Japan}

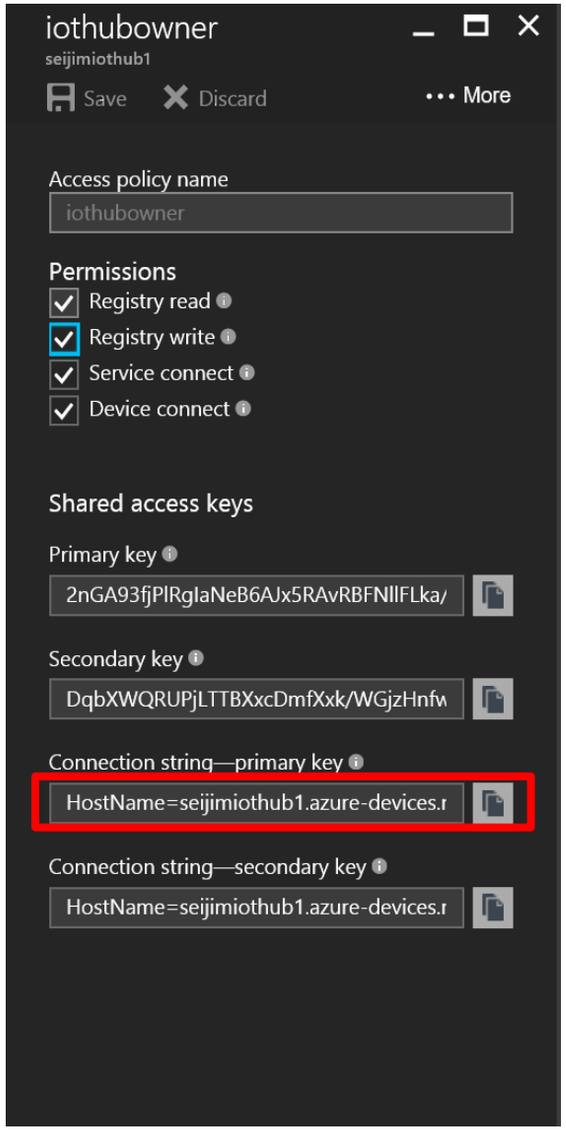


T02_03 : Confirmation after creating IoT Hub

Select from the Azure portal as follows

- [IoT Hub] → [Applicable IoT Hub] → [Shared access policy] → [iothubowner]
- Save [Connection string - primary key] in Notepad etc.

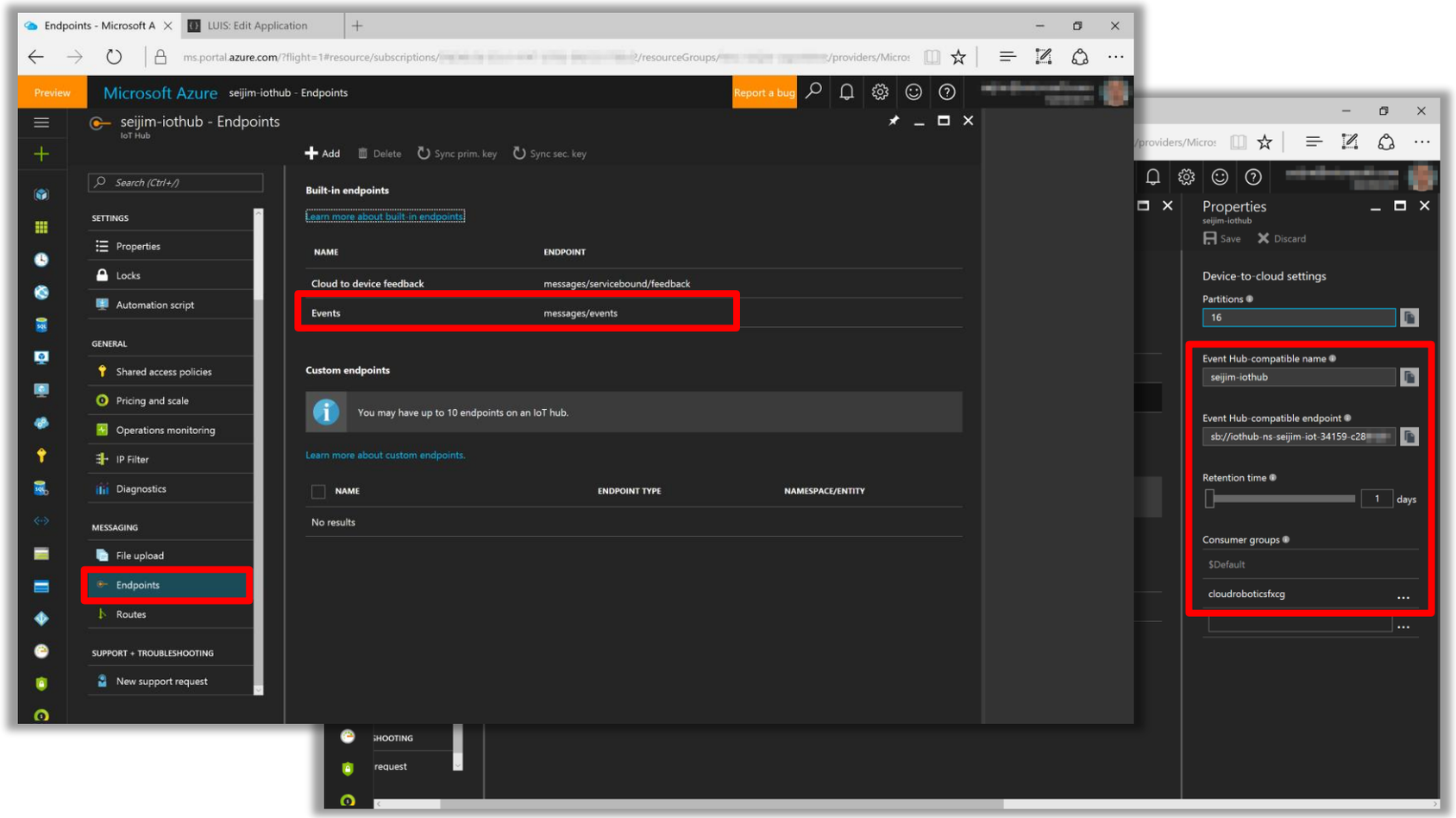




T02_04 : Confirmation and setting after creation of IoT Hub

Select from the Azure portal as follows

- [IoT Hub] → [Applicable IoT Hub] → [MESSAGING] → [Endpoints]
- Save [Event hub-compatible name] and [Event hub-compatible end point] in Notepad etc.
- Set [Consumer group] = {cloudroboticsfxcg} and press the [Save] button



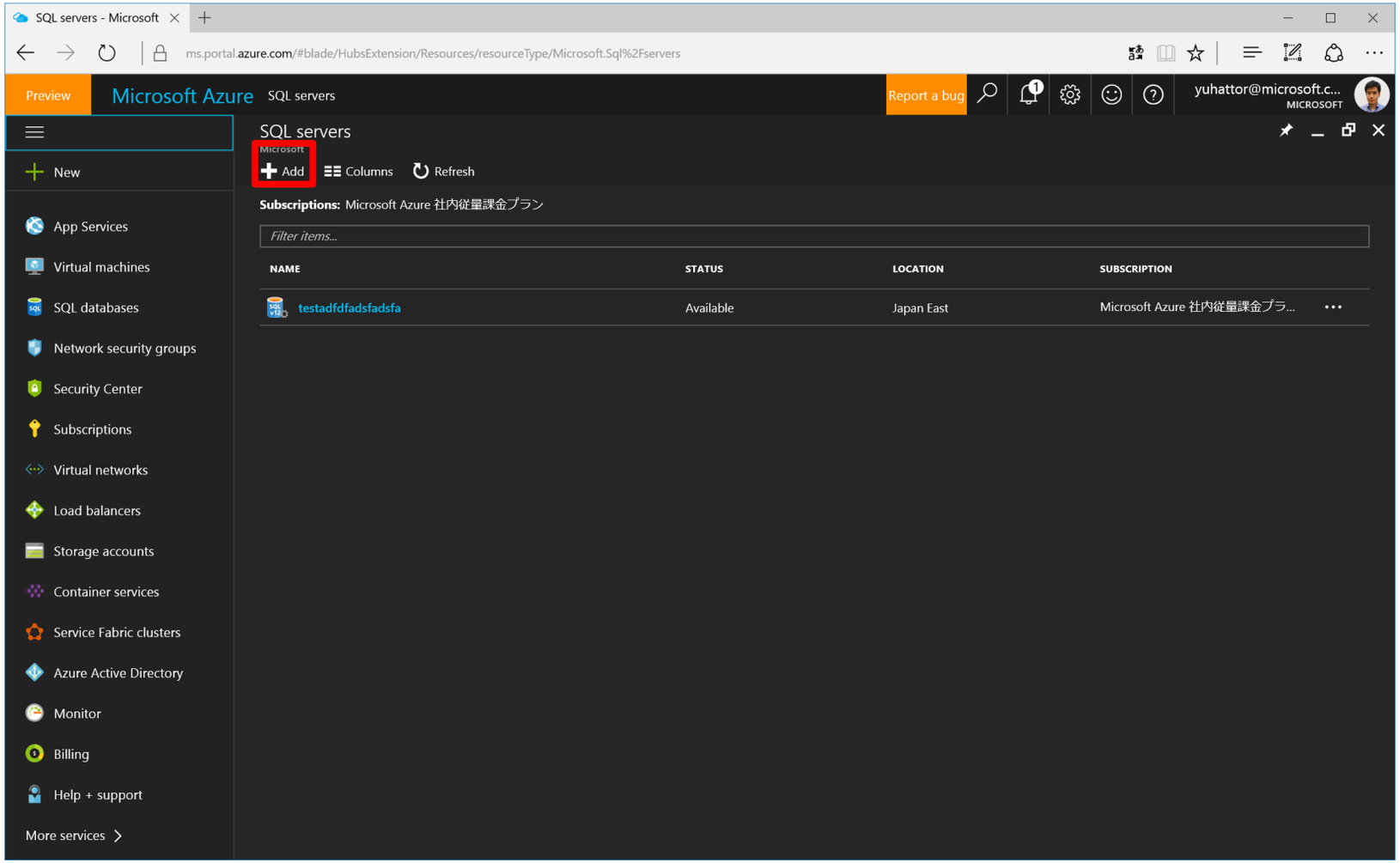
Tutorial 03

Create SQL Database

T03_01 : Create SQL Server

Select from the Azure portal as follows

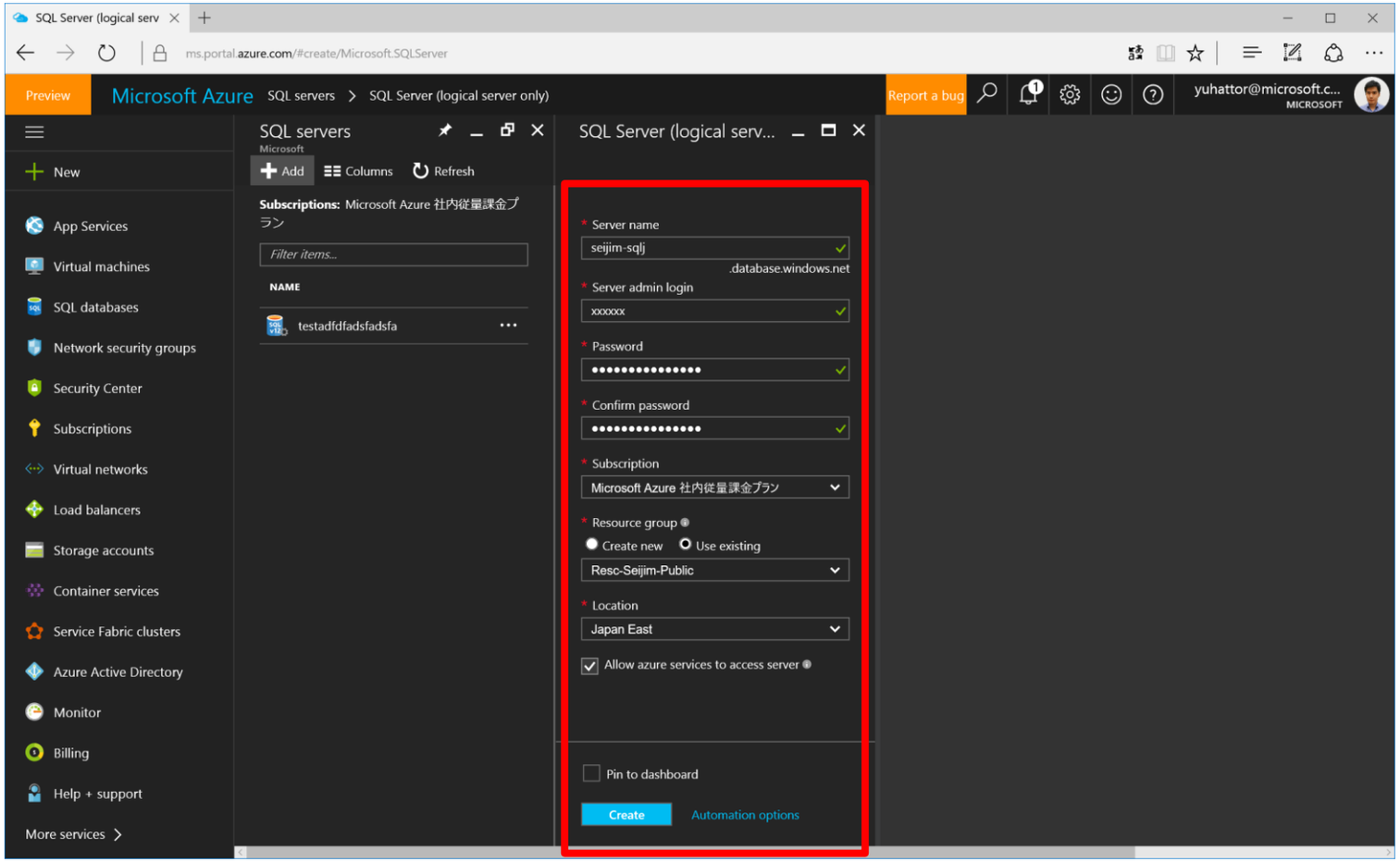
- [SQL Servers]→[+Add]



T03_02 : Input parameters for SQL Server

Set the parameters as below and press the [Create] button

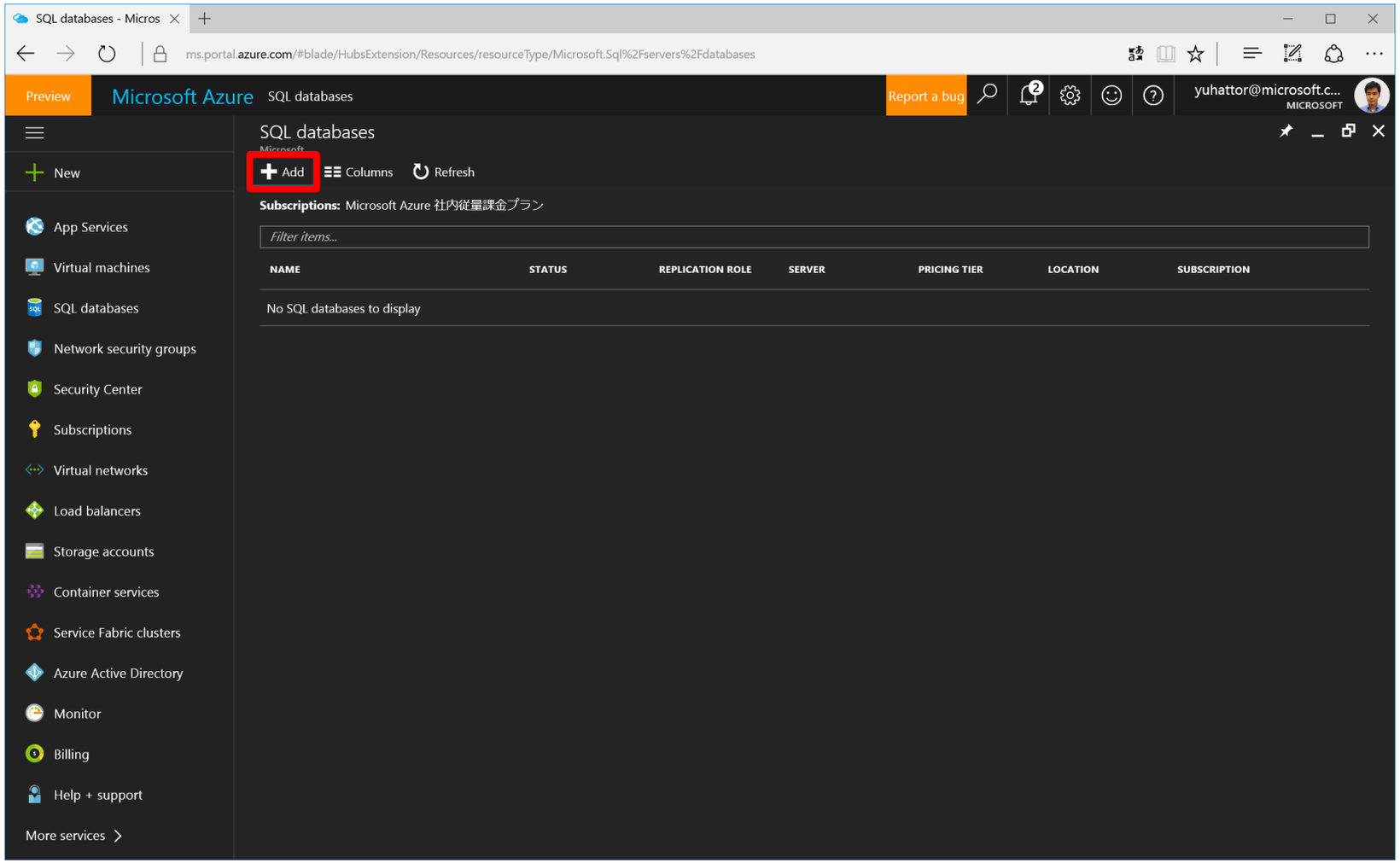
- [Server name] = {Set lowercase letters without duplication}
- [Server admin login] = {Set administrative UID}
- [Password] = {Set complex character string}
- [Resource group] = {Set value of T 01}
- [Location] = {East Japan or West Japan}



T03_03 : Create SQL Database

Select from the Azure portal as follows

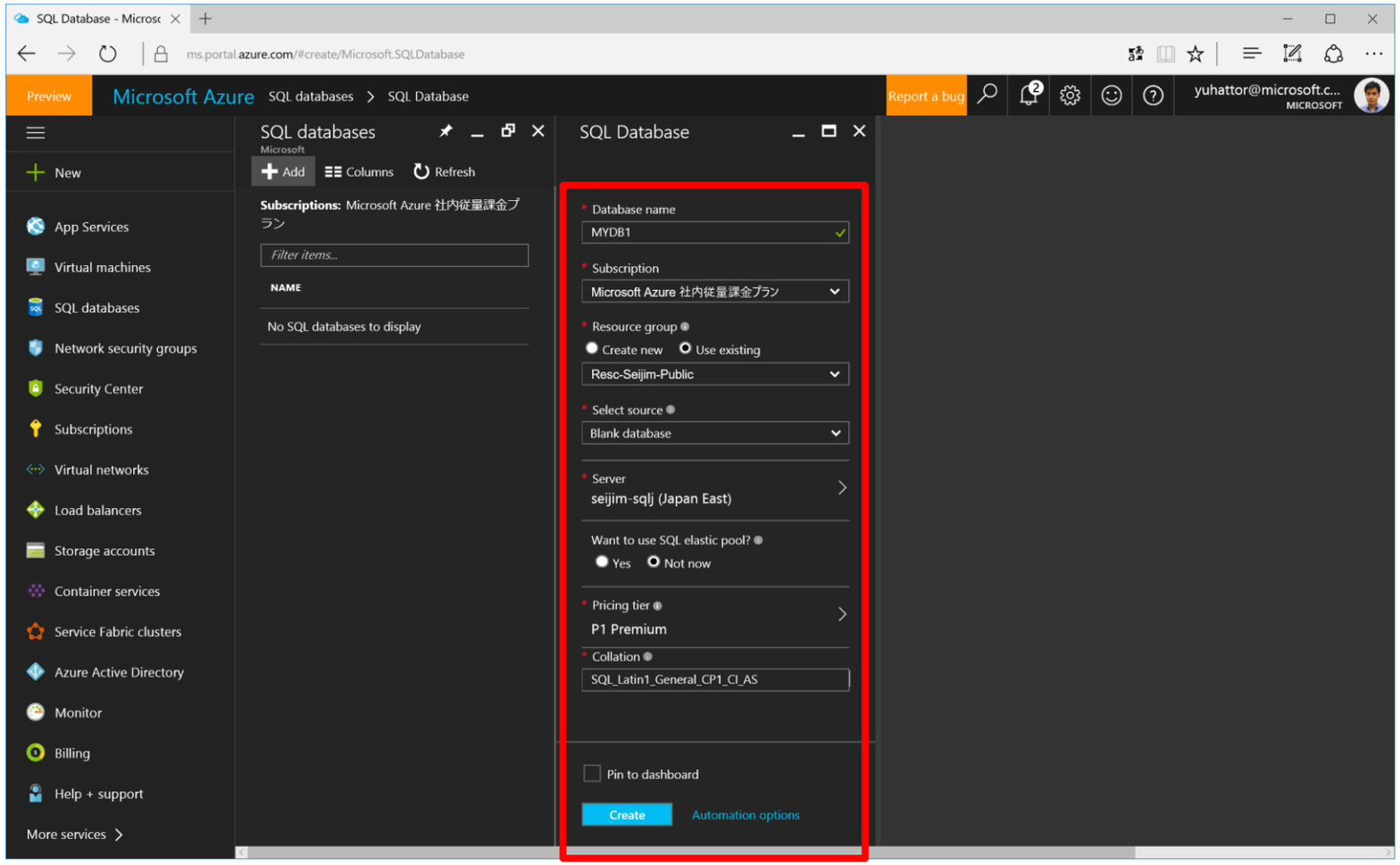
- [SQL databases]→[+Add]



T03_04 : Input parameters for SQL Database

Set the parameters as below and press the [Create] button

- [Database name] = {Set with easy-to-manage names}
- [Resource group] = {Set value of T01}
- [Server] = {Select SQL Server you created}
- [Price level] = {P1 Premium} (*)
- [Collation order]
  = {SQL_Latin1_General_CP1_CI_AS}



* In this hands-on, there is a table (RBApp.IoTHubLog) with clustering column store index. If you delete the index, you can change the price level to Standard.

* Using the column store index generally enables data compression of 10 to 15 times and improves the DWH query performance by up to several hundred times.

* Price Level: With Premium, in-memory OLTP function can be used. It makes performance up to 30 times better.

Reference material

Details per price level

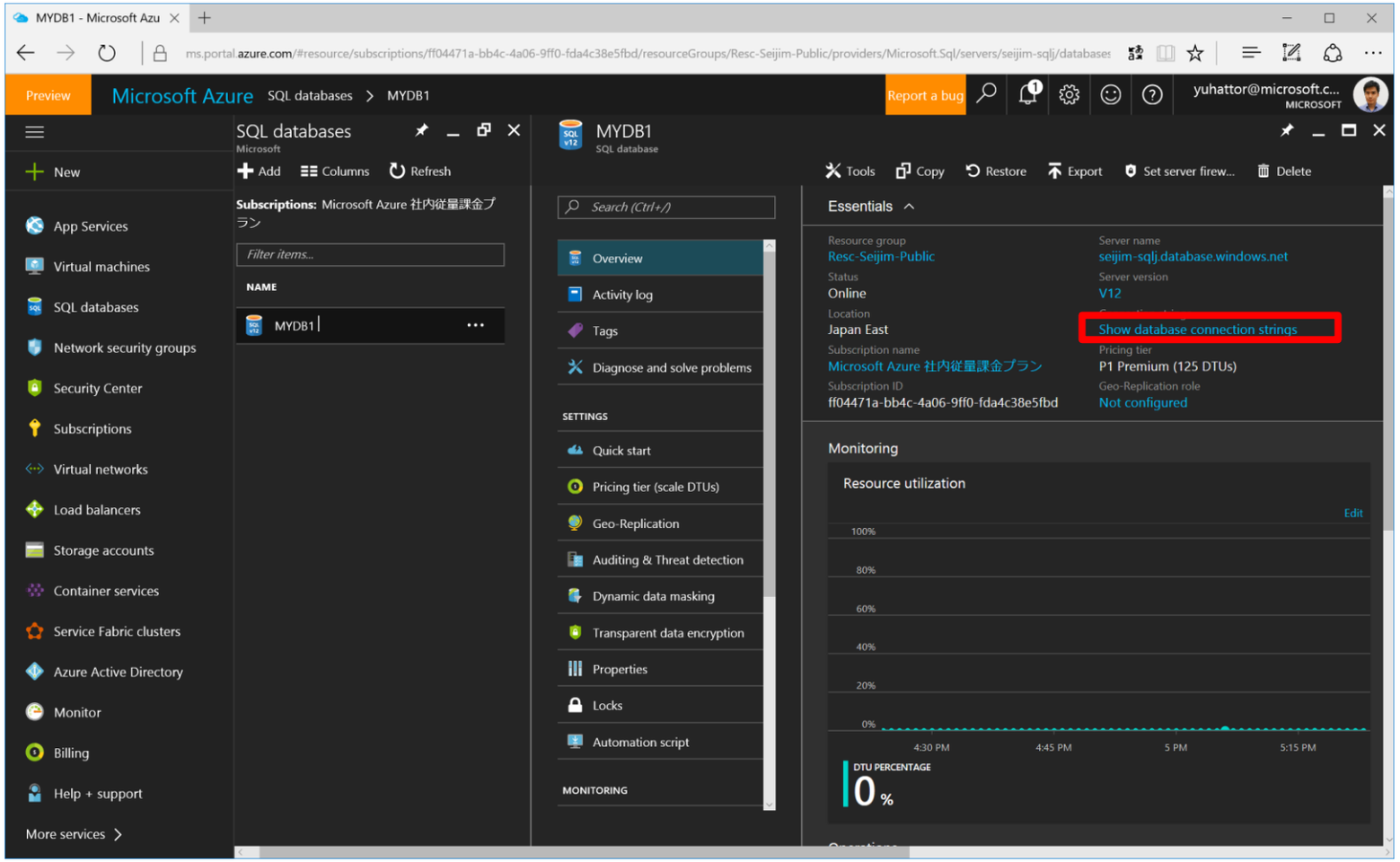
機能	Basic	Standard				Premium				
		S0	S1	S2	S3*	P1	P2	P4	P6	P11
Usable time SLA	99.99%	99.99%				99.99%				
Maximum DB size	2 GB	250 GB				500 GB				1 TB
DTU (Database Throughput Units)	5	10	20	50	100	125	250	500	1,000	1,750
Point In Time Restore	Any time during the past 7 days	Any time during the past 35 days				Any time during the past 35 days				
Disaster Recovery	Geographic restore	Standard geographic replication (offline secondary)				Active geographic replication (up to 4 [readable] online secondary backups				
Performance Target	Transactions per hour	Transactions per minute				Transactions per second				
	d	521/m	934/m	2,570/m	5,100/m	105/s	228/s	447/s	735/s	1,286/s
Predictability	Good (in hours)	High (in seconds)				Highest (in seconds)				
TPM conversion value	276	521	934	2,570	5,100	6,300	13,680	26,820	44,100	77,175
Maximum number of concurrent requests	30	60	90	120	200	200	400	800	1,600	2,800
Maximum number of sessions	300	600	900	1,200	2,400	2,400	4,800	9,600	19,200	33,600
Price per hour	¥0.78/h	¥2.33/h	¥4.66/h	¥11.65/h	¥23.28/h	¥72.05/h	¥144.08/h	¥288.66/h	¥576.30/h	¥1086.30/h
Price per month	¥612/mon	¥1,734/mon	¥3,468/mon	¥8,670/mon	¥17,340/mon	¥53,550/mon	¥107,202/mon	¥214,812/mon	¥428,808/mon	¥808,248/mon

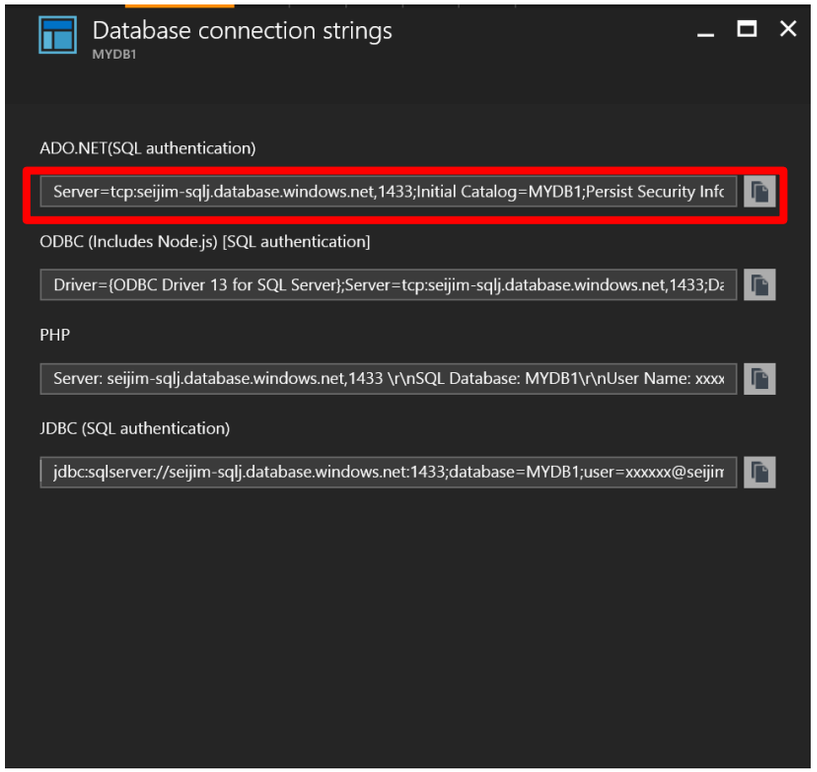
Upgrade Azure SQL Database Web or Business Databases to New Service Tiers
http://azure.microsoft.com/en-us/documentation/articles/sql-database-upgrade-new-service-tiers/
Azure SQL Database Pricing
http://azure.microsoft.com/en-us/pricing/details/sql-database/

T03_05 : Confirmation after SQL Database creation

Select from the Azure portal as follows

- [SQL database] → [Applicable DB] → [Show database connection strings]
- Copy the connection string of [ADO.NET (SQL authentication)] to Notepad etc., and edit the user ID / password and save it





Tutorial 04

Create table on SQL Database

DB table definition

Cloud Robotics FX - SQLDB : Device definition 1

```
CREATE TABLE [RBFX].[DeviceMaster]
(
    [SeqId] int IDENTITY NOT NULL,
    [DeviceId] NVARCHAR(100) NOT NULL,
    [DeviceType] NVARCHAR(100) NULL,
    [Status] VARCHAR(20) NULL,
    [ResourceGroupId] VARCHAR(40) NULL,
    [Description] NVARCHAR(1000) NULL,
    [Registered_DateTime] DATETIME NULL,
CONSTRAINT [PK_DeviceMaster] PRIMARY KEY CLUSTERED
(
    [DeviceId] ASC
))

CREATE TABLE [RBFX].[DeviceRouting]
(
    [SeqId] int IDENTITY NOT NULL,
    [DeviceId] NVARCHAR(100) NOT NULL,
    [RoutingKeyword] NVARCHAR(100) NOT NULL,
    [TargetType] VARCHAR(20) NULL,
    [TargetDeviceGroupId] NVARCHAR(100) NULL,
    [TargetDeviceId] NVARCHAR(100) NULL,
    [Status] VARCHAR(20) NULL,
    [Description] NVARCHAR(1000) NULL,
    [Registered_DateTime] DATETIME NULL,
CONSTRAINT [PK_DeviceRouting] PRIMARY KEY CLUSTERED
(
    [DeviceId] ASC,
    [RoutingKeyword] ASC
))
```

[RBFX].[DeviceMaster] : Device master information table

- → [DeviceId] Column is registered with the same value as DeviceId registered in IoT Hub
- → Store device type and information
- → Values of the [Status] column are "Active" or "NA". Only "Active" rows are targeted
- → Values of the [ResourceGroupId] column means the contract section of the customer In the mechanism of SaaS
- → Do not perform routing for devices that are not registered in this master.

[RBFX].[DeviceRouting] : Device routing definition table

- → Device or DeviceGroup to be routed is identified by the source [DeviceId]
- → Routing is determined by [DeviceId] + [RoutingKeyword]
- → Values of the [TargetType] column are"Device" or "DeviceGroup". The destination column is specified by its value
- → Values of the [Status] column are "Active" or "NA". Only "Active" rows are targeted

Cloud Robotics FX - SQLDB : Device definition 2

```
CREATE TABLE [RBFX].[DeviceGroup]
(
    [SeqId] int IDENTITY NOT NULL,
    [DeviceGroupId] NVARCHAR(100) NOT NULL,
    [DeviceId] NVARCHAR(100) NOT NULL,
    [Registered_DateTime] DATETIME NULL,
CONSTRAINT [PK_DeviceGroup] PRIMARY KEY CLUSTERED
(
    [DeviceGroupId] ASC,
    [DeviceId] ASC
))

CREATE INDEX [IDX1] ON [RBFX].[DeviceGroup]
(
    [DeviceId] ASC
)
```

[RBFX].[DeviceGroup] : Device group information table

→ The [DeviceGroupId] column has the same value as [TargetDeviceGroupId] in the [RBFX]. [DeviceRouting] table

→ The [DeviceId] column has the same value as the DeviceId registered in the IoT Hub

Cloud Robotics FX - SQLDB : Application definition

```
CREATE TABLE [RBFX].[AppMaster]
(
   [SeqId] int IDENTITY NOT NULL,
   [AppId] NVARCHAR(100) NOT NULL,
   [StorageAccount] NVARCHAR(256) NULL,
   [StorageKeyEnc] VARBINARY(2000) NULL,
   [AppInfoEnc] VARBINARY(8000) NULL,
   [AppInfoDeviceEnc] VARBINARY(8000) NULL,
   [Status] VARCHAR(20) NULL,
   [Description] NVARCHAR(1000) NULL,
   [Registered_DateTime] DATETIME NULL,
CONSTRAINT [PK_AppMaster] PRIMARY KEY CLUSTERED
(
   [AppId] ASC
))

CREATE TABLE [RBFX].[AppRouting]
(
   [SeqId] int IDENTITY NOT NULL,
   [AppId] NVARCHAR(100) NOT NULL,
   [AppProcessingId] NVARCHAR(100) NOT NULL,
   [BlobContainer] NVARCHAR(100) NULL,
   [FileName] NVARCHAR(100) NULL,
   [ClassName] NVARCHAR(100) NULL,
   [Status] VARCHAR(20) NULL,
   [DevMode] VARCHAR(5) NULL,
   [DevLocalDir] NVARCHAR(1000) NULL,
   [Description] NVARCHAR(1000) NULL,
   [Registered_DateTime] DATETIME NULL,
CONSTRAINT [PK_AppRouting] PRIMARY KEY CLUSTERED
(
   [AppId] ASC,
   [AppProcessingId] ASC
))
```

[RBFX].[AppMaster] : Application (DLL) storage destination information table

- → Configure by application system unit
- → Store Storage Account and Key of the application (DLL) storage location, connection information for the application, etc.
- → [StorageKeyEnc] column : Encrypt Storage Key with SQL encryption function
- → [AppInfoEnc] column : Encrypt cloud side application specific DB connection information, etc. with SQL encryption function
- → [AppInfoDeviceEnc] column : Encrypt the device-side application-specific DB connection information, etc. with the SQL encryption function
- → The value of [Status] column is ”Active” or “NA”. Only "Active" rows are targeted

[RBFX].[AppRouting] : Application (DLL) routing definition table

- → Mapping application (DLL) from [AppId] and [AppProcessingId] of electronic message (JSON) sent from device
- → Application (DLL) storage destination information is acquired from [RBFX]. [AppRoutingMaster] table and [BlobContainer] column
- → Set the value of the [ClassName] column to <namespace>. <Class name>
- → The value of [Status] column is ”Active” or “NA”. Only "Active" rows are targeted
- → When the value in the [DevMode] column is "True", it loads from [DevLocalDir] without checking the storage location of the BLOB
  - → This is Used for testing the DLL unit test on a local PC.

Cloud Robotics FX - SQLDB : Customer system definition (for SaaS)

```
CREATE TABLE [RBFX].[CustomerInfo]
(
   [SeqId] int IDENTITY NOT NULL,
   [CustomerId] NVARCHAR(100) NOT NULL,
   [CustomerName] NVARCHAR(100) NULL,
   [Description] NVARCHAR(1000) NULL,
   [Registered_DateTime] DATETIME NULL,
CONSTRAINT [PK_CustomerInfo] PRIMARY KEY CLUSTERED
(
   [CustomerId] ASC
))

CREATE TABLE [RBFX].[CustomerResource]
(
   [SeqId] int IDENTITY NOT NULL,
   [CustomerId] NVARCHAR(100) NOT NULL,
   [ResourceGroupId] NVARCHAR(40) NOT NULL,
   [ResourceGroupName] NVARCHAR(100) NULL,
   [SqlConnectionStringEnc] VARBINARY(2000) NULL,
   [Description] NVARCHAR(1000) NULL,
   [Registered_DateTime] DATETIME NULL,
CONSTRAINT [PK_CustomerResource] PRIMARY KEY CLUSTERED
(
   [CustomerId] ASC,
   [ResourceGroupId] ASC
))

CREATE INDEX [IDX1] ON [RBFX].[CustomerResource]
(
   [ResourceGroupId] ASC
)
```

[RBFX].[CustomerInfo] : Customer information table

[RBFX].[CustomerResource] : Customer resource management table

→ Resource group = Managing pepper and devices in resource groups in contract units, such as departments

→ The activation of the application (DLL) is also managed by this resource group unit

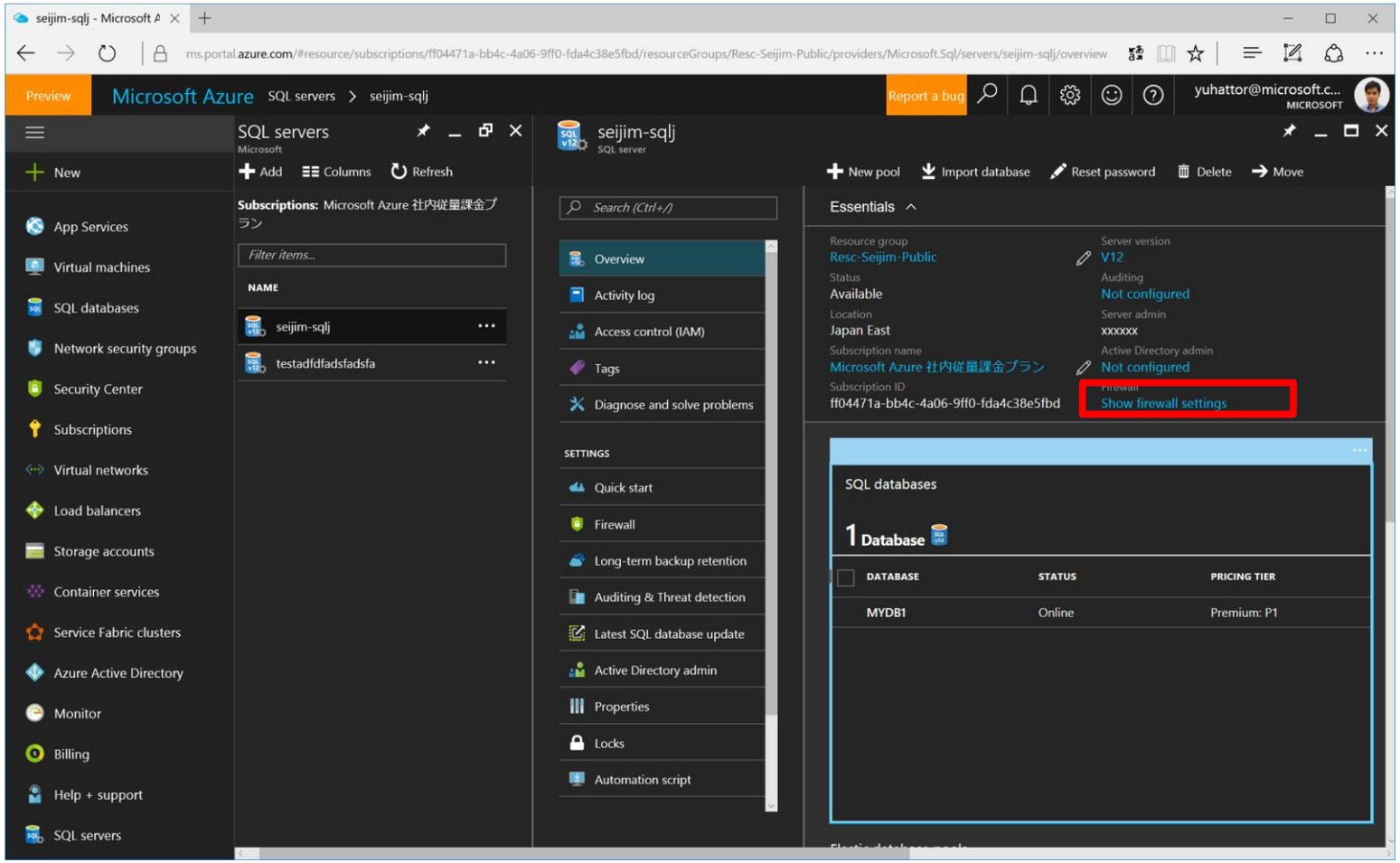
→ The value of the [ResourceGroupId] column is a GUID value with a hyphen

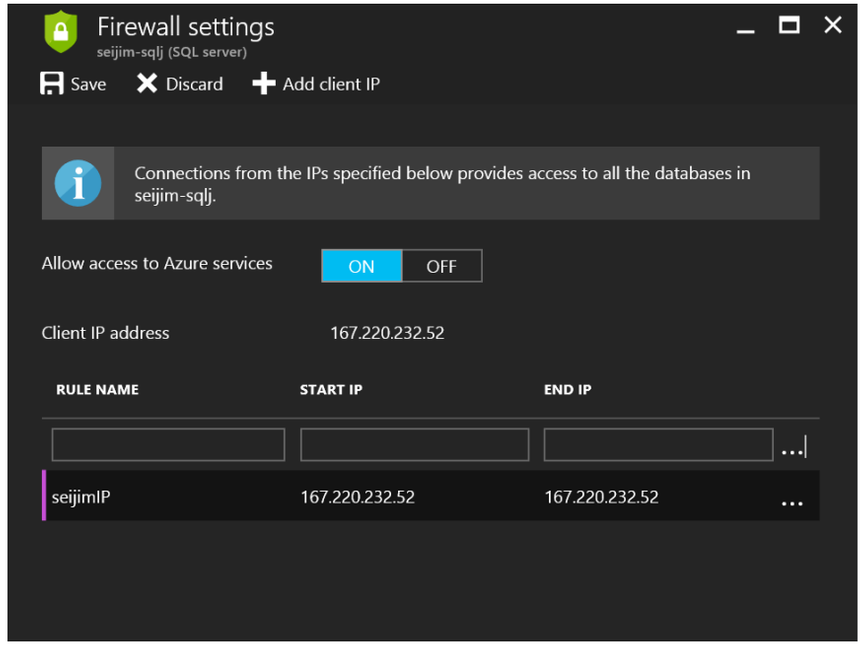
→ The [SqlConnectionStringEnc] column is the connection information of the DB for each customer and is encrypted with the SQL encryption function

T04_01 : SQL Server Firewall Setting

Select from the Azure portal as follows

- [SQL Server] → [Applicable server] → [Show firewall settings]
- Press [+Add Client IP], give it a name and press [Save]

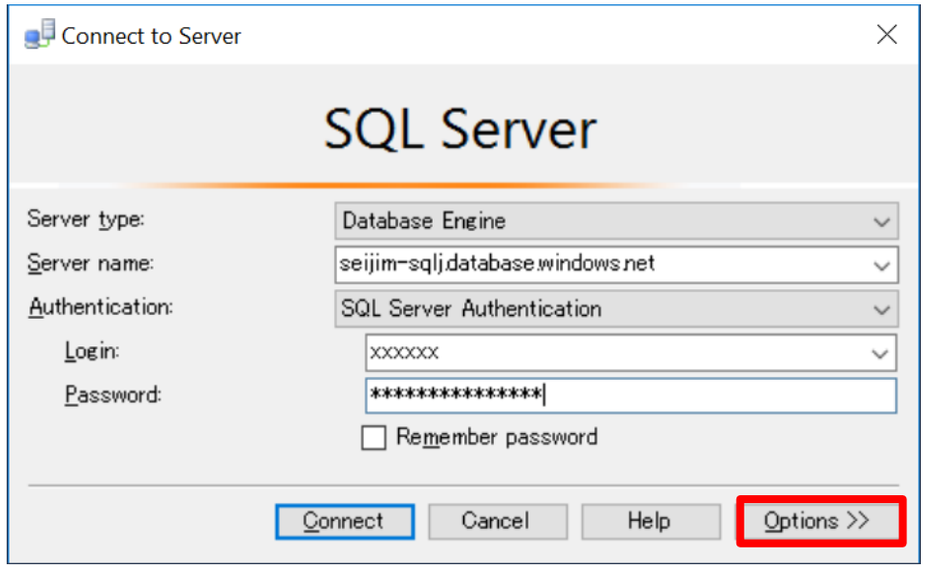


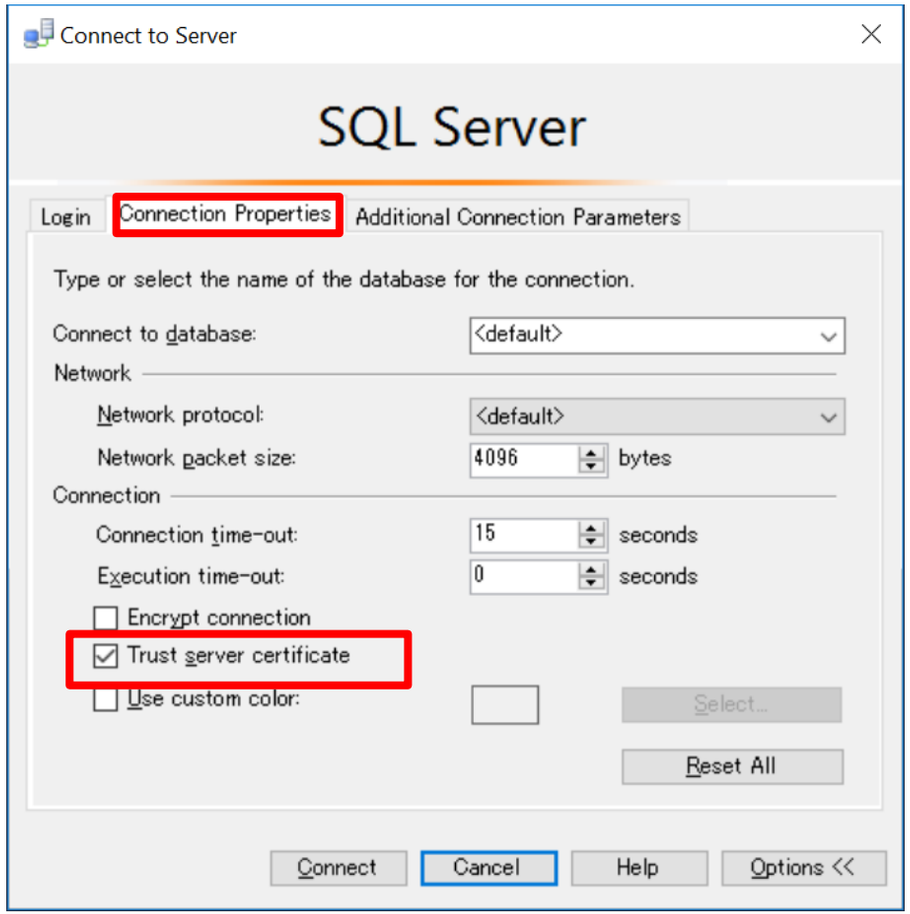


T04_02 : Starting SQL Server Management Studio

Start up the SQL Server Management Studio you installed in advance
Paste the Server name from the ADO.NET connection string saved in Notepad

- Select [SQL Server Authentication] and enter the administrator user ID and password
- Press [Options], and check [Trust server certificate] on the [Connection Properties] tab
- Press the [Connect] button

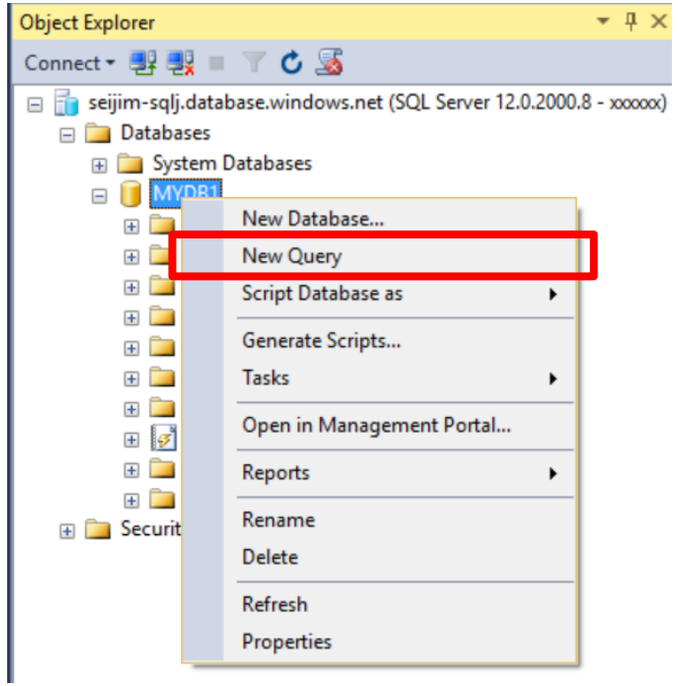


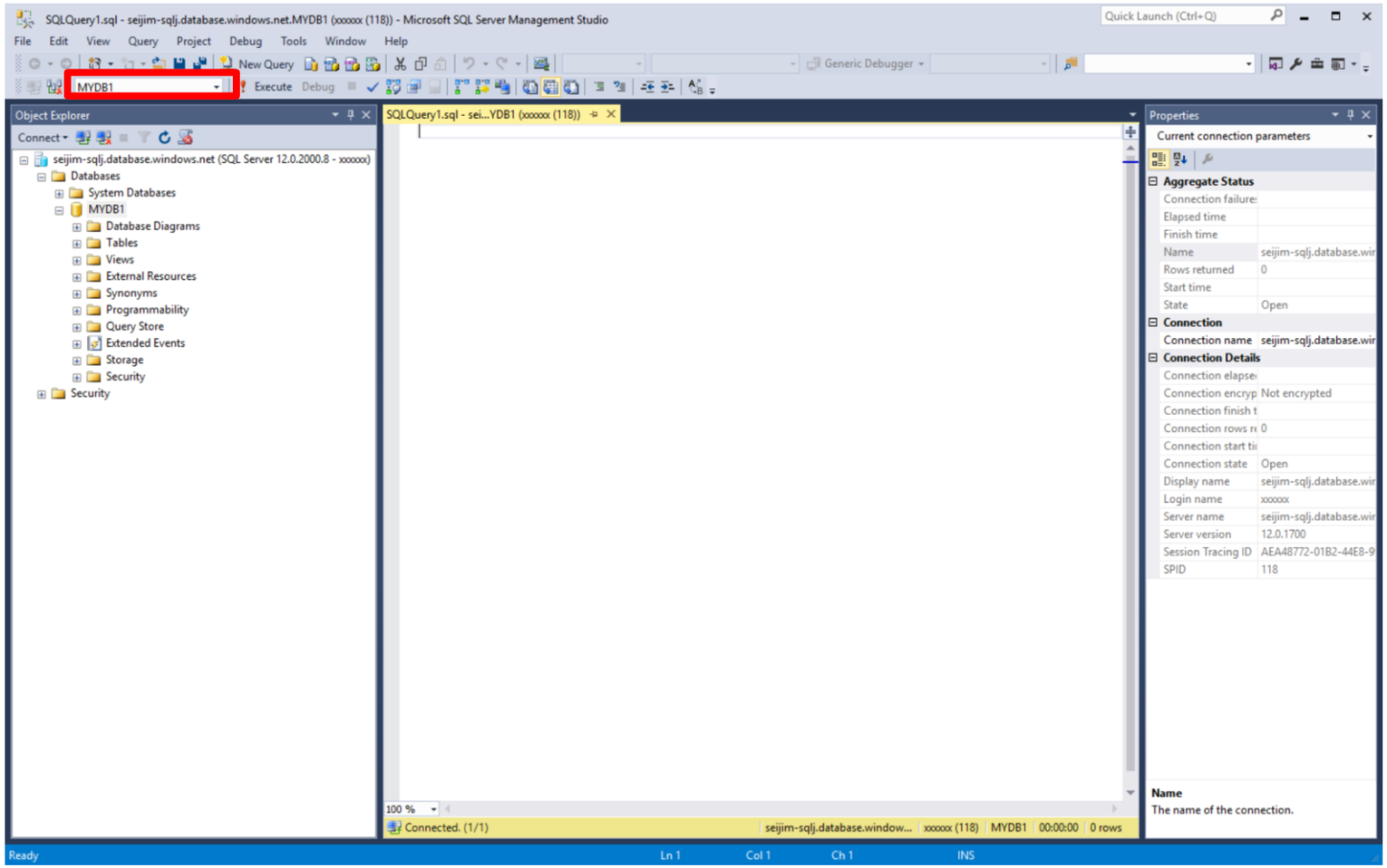


T04_03 : Working with SQL Server Management Studio

Select the target database in SQL Server Management Studio

- Select the relevant DB from the Object Explorer and open menu and choose [New Query]
- Make sure relevant DB is selected on the tool bar

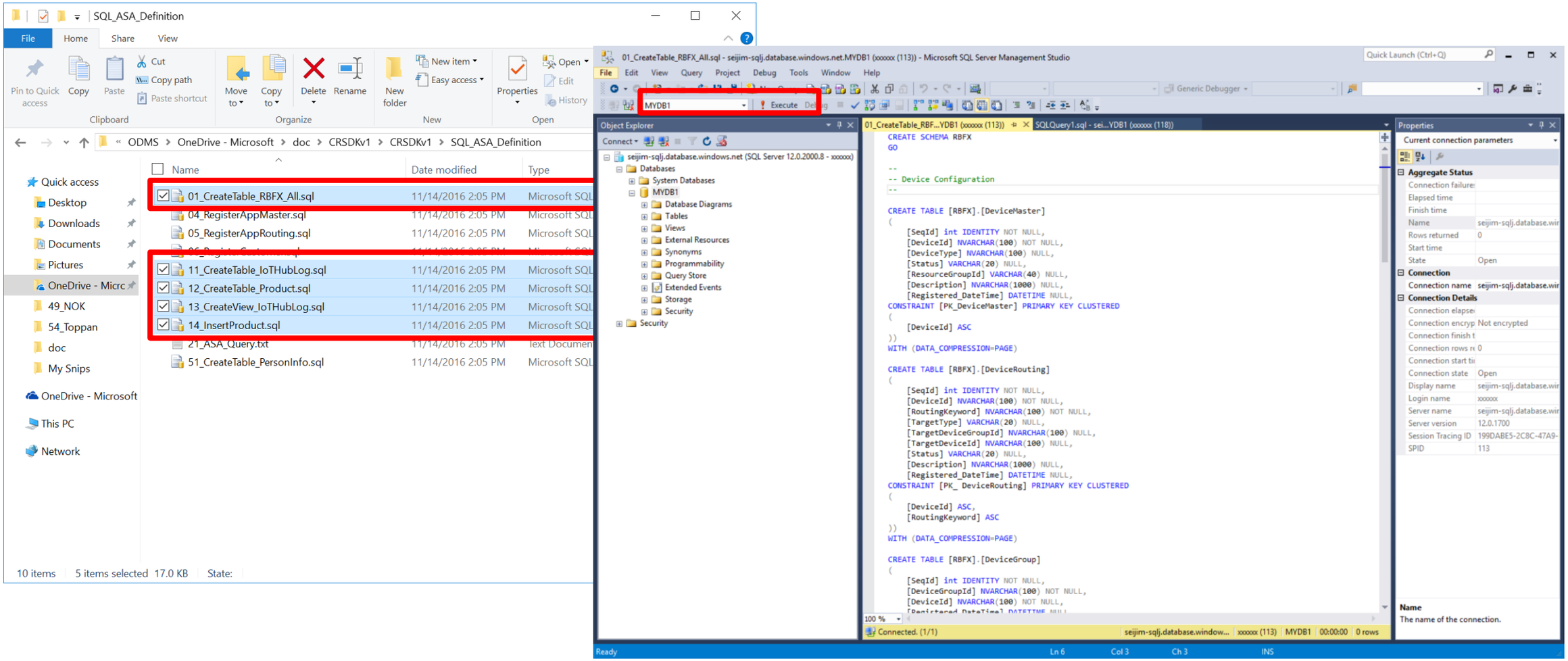




T04_04 : Execution of SQL script file

Run SQL scripts from SQL Server Management Studio and create tables for Cloud Robotics SDK and applications

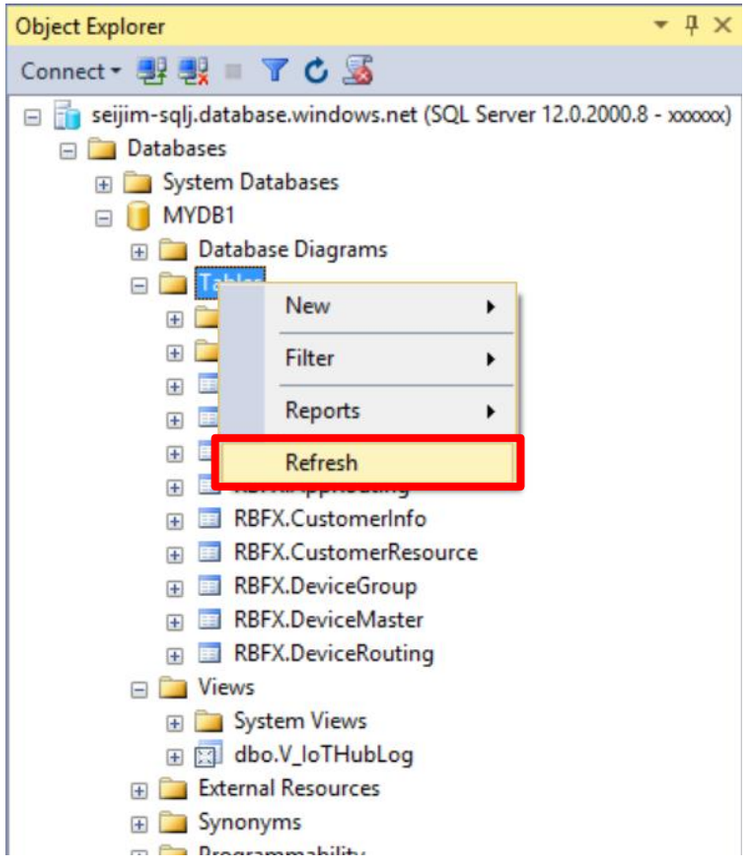
- Extract the "CRSDKv1.zip" you downloaded and go to "CRSDKv1¥SQL_ASA_Definition"
- Double-click the SQL file and execute from the Management Studio to the relevant DB

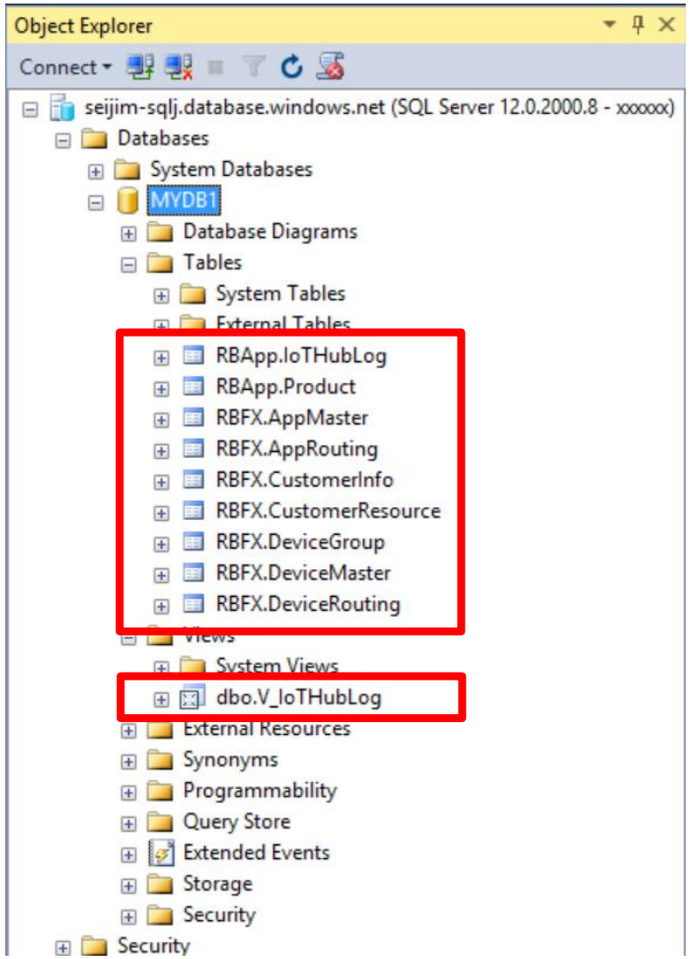


T04_05 : Confirm the table

Confirm that the table was successfully created

- Right-click on "Table" in Object Explorer and press "Refresh"
- If the enclosed table is created below, confirmation is completed





Tutorial 05

Build and set up Cloud Robotics SDK

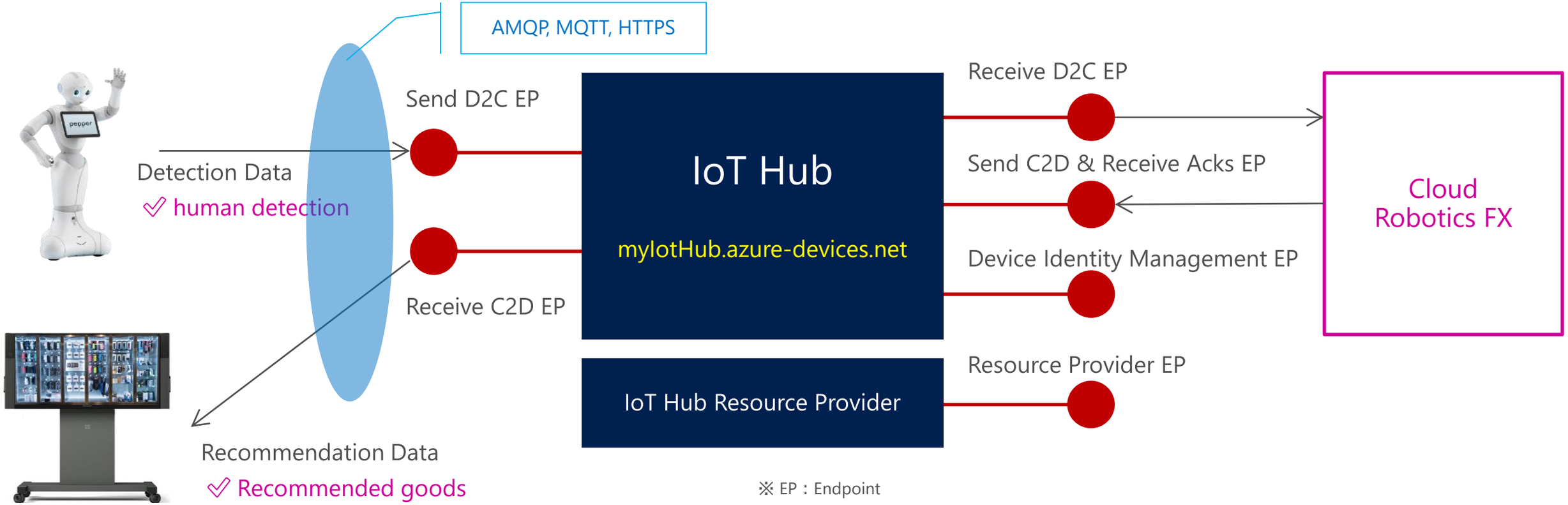
参考資料

Communication with Azure IoT Hub

Reliable two-way communication between device and backend

- ✓ communication security
- ✓ Scalability of millions of device connections
- ✓ Multiplatform*1 / Multi language*2 SDK

*1 ：RTOS, Linux, Windows, ...etc.
*2 ：C, .NET, Java, Node.js



※ EP：Endpoint

※reference: https://azure.microsoft.com/ja-jp/documentation/articles/iot-hub-devguide/

Communication format

Cloud Robotics FX - D2C(↑) & C2D(↓) Control Format

```
{
  "RbHeader": {
    "RoutingType": "CONTROL",
    "AppId": "PepperShopApp",
    "AppProcessingId": "ReqAppInfo",
    "MessageSeqno": "201",
    "SendDateTime": "2016-04-01 14:05:22.038"
  },
  "RbBody": {
    "visitor": "u001"
  }
}
```

RoutingType : required
→ “CONTROL” fixed
AppId : required
→ Specific use of application system
AppProcessingId : required
→ “ReqAppInfo” (Application information acquisition)
MessageSeqno : required
→ Set by the instructing application
SendDateTime : required
→ Local time on sending (up to milliseconds)

```
{
  "RbHeader": {
    "RoutingType": "CONTROL",
    "AppId": "PepperShopApp",
    "AppProcessingId": "ResAppInfo",
    "MessageSeqno": "202",
    "SendDateTime": "2016-04-01 14:05:23.425"
  },
  "RbBody": {
    <AppInfoDevice Column of RBFX.AppMaster>
  }
}
```

RoutingType : “CONTROL” fixed

AppId : : Content set by the caller

AppProcessingId : “ResAppInfo” (responce to application information acquisition)

MessageSeqno : Content set by the caller

SendDateTime : Content set by the caller

Cloud Robotics FX - D2C(↑) & C2D(↓) App Format

```
{
  "RbHeader": {
    "RoutingType": "D2D",
    "RoutingKeyword": "Default",
    "AppId": "PepperShopApp",
    "AppProcessingId": "",
    "MessageId": "MSG01",
    "MessageSeqno": "159",
    "SendDateTime": "2016-04-01 14:05:22.038",
    "SourceDeviceId": "",
    "SourceDeviceType": "",
    "SourceDevRescGroupId": "",
    "TargetType": "",
    "TargetDeviceGroupId": "",
    "TargetDeviceId": "",
    "ProcessingStack": ""
  },
  "RbBody": {
      ...<Free settings for application>...
  }
}
```

RoutingType : required
- → Determine the behavior of Robotics FX's device routing and calling an application (DLL)
  - ・Device routing is required: "D2D"
  - ・Only the application (DLL) call is necessary: "CALL",
  - ・For a simple logging message: "LOG"

RoutingKeyword : Optional
- → Device routing is by SourceDeviceId + RoutingKeyword. "Default" is automatically set when not specified

AppId : required
- → specific use of application system

AppProcessingId : "CALL" required / "D2D" Optional / "LOG" Unnecessary
- → For setting, Robotics FX specifies an application (DLL) with <AppId> + <AppProcessingId> from RBFX.AppRouting and calls it

MessageId : Optional
- → Freely use within application (DLL) or "D2D" cooperation

MessageSeqno : required
- → Set communication serial number. It is necessary for tracking when a problem occurs with Robotics FX or an application (DLL)

SendDateTime : required
- → Local time on sending (up to milliseconds)

SourceDeviceId : Unnecessary
- → Robotics FX automatically sets the device ID of IoT Hub

SourceDeviceType : Unnecessary
- → Robotics FX specifies the type by <SourceDeviceId> from RBFX.DeviceMaster and set automatically

SourceDevRescGroupId : Unnecessary
- → Robotics FX specifies the id by <SourceDeviceId> from RBFX.DeviceMaster and automatically set the value of <ResourceGroupId>

TargetType : Can be overwritten with application (DLL)
- → Values must be "Device" or "DeviceGroup"
- → Robotics FX specifies the destination using <SourceDeviceId> + <RoutingKeyword> from RBFX.DeviceRouting and automatically sets

TargetDeviceGroupId : Can be overwritten with application (DLL)
- → For simultaneous transmission to multiple devices
- → Robotics FX specifies the destination using <SourceDeviceId> + <RoutingKeyword> from RBFX.DeviceRouting and automatically sets

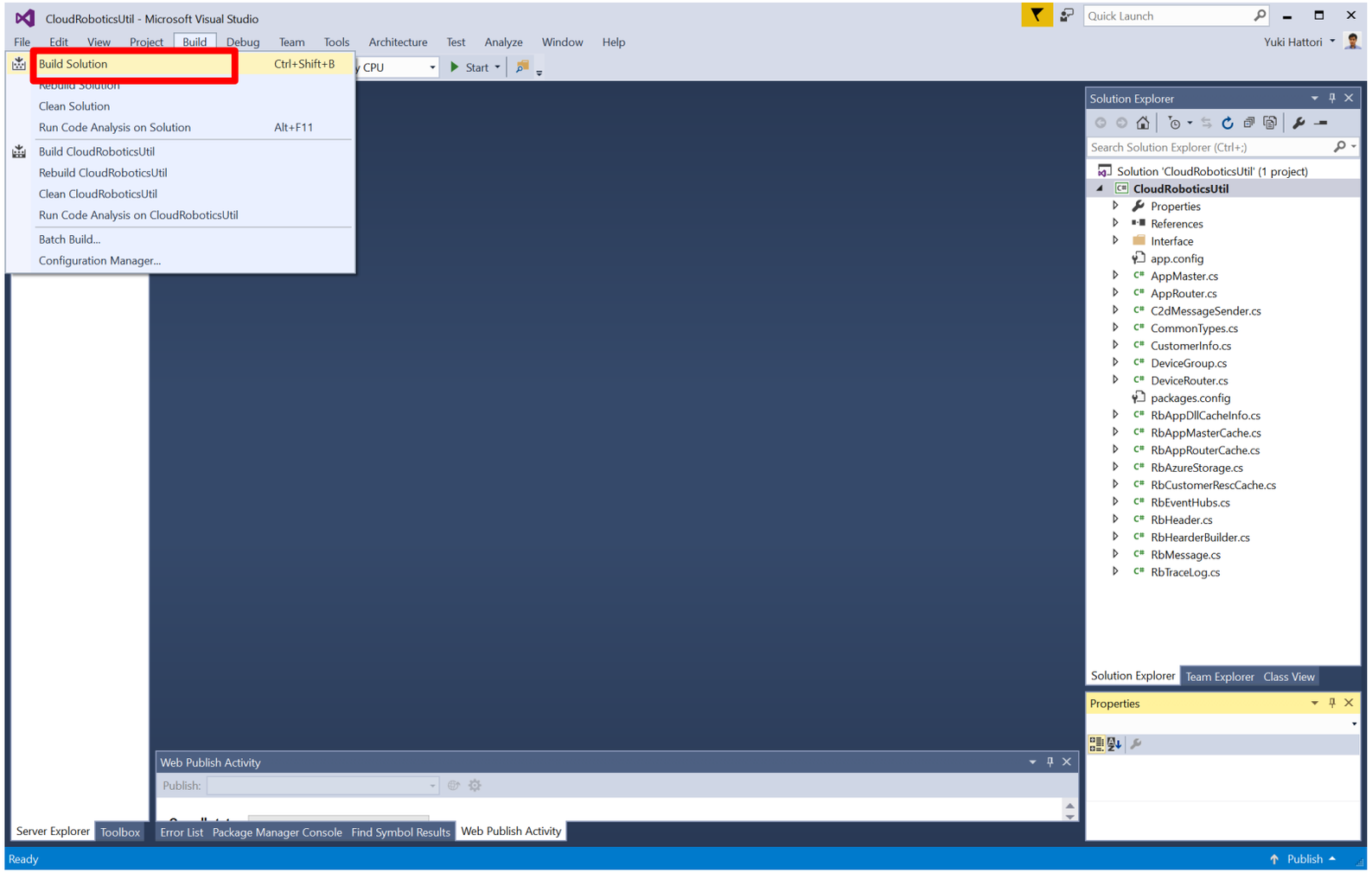
TargetDeviceId : Can be overwritten with application (DLL)
- → For sending to a single device
- → Robotics FX specifies the destination using <SourceDeviceId> + <RoutingKeyword> from RBFX.DeviceRouting and automatically sets

ProcessingStack : Unnecessary
- → Automatically setting the result of which DLL the Robotics FX called and contents at the time of exception occurrence

T05_01 : Build CloudRoboticsUtil.dll

Build Cloud Robotics Utility (common function DLL)

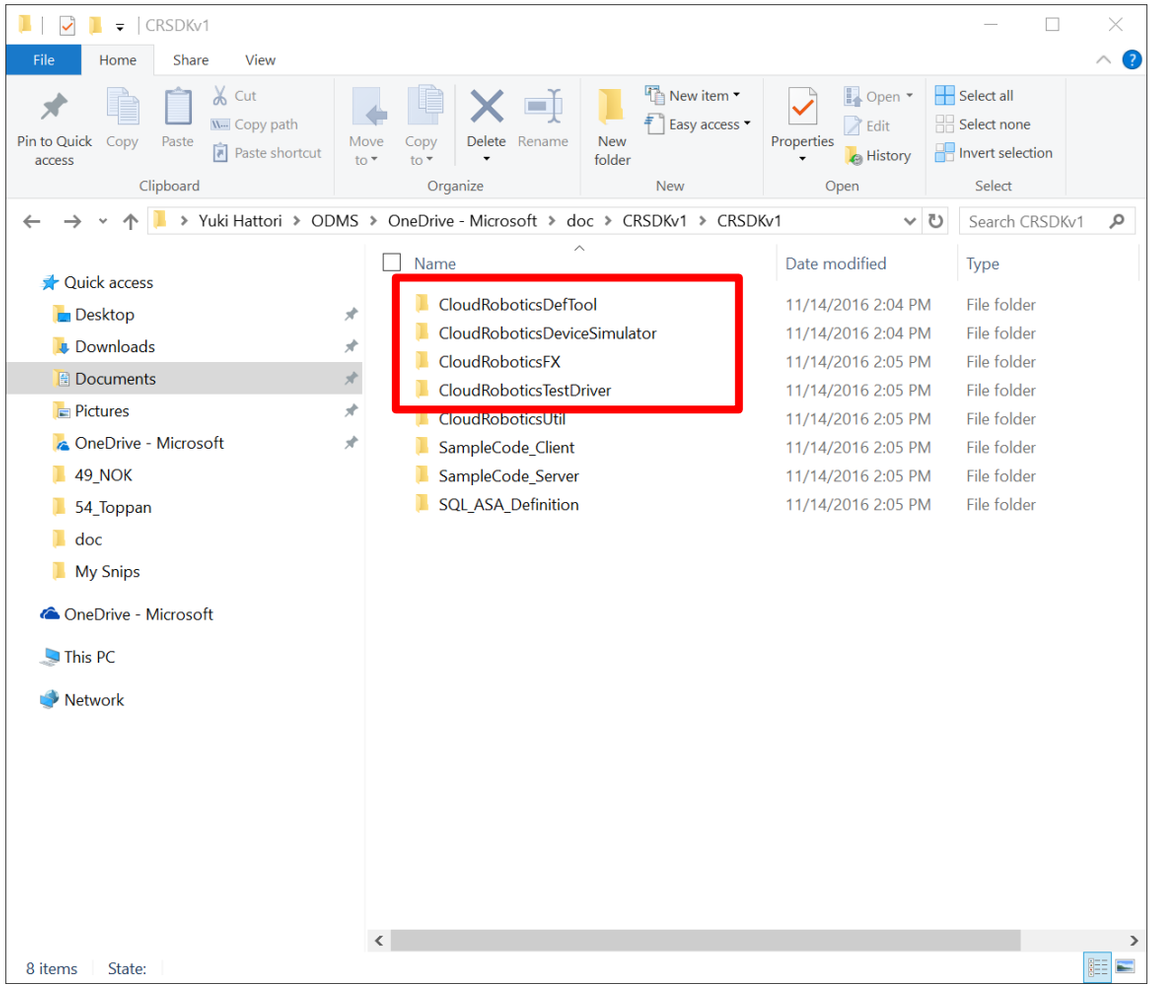
- Open the "CRSDKv1¥CloudRoboticsUtil" folder and double click "CloudRoboticsUtil.sln"
- From the menu bar of Visual Studio 2015, select [Build] → [Build Solution]



T05_02 : Build each solution file

Similarly, build the following solution file

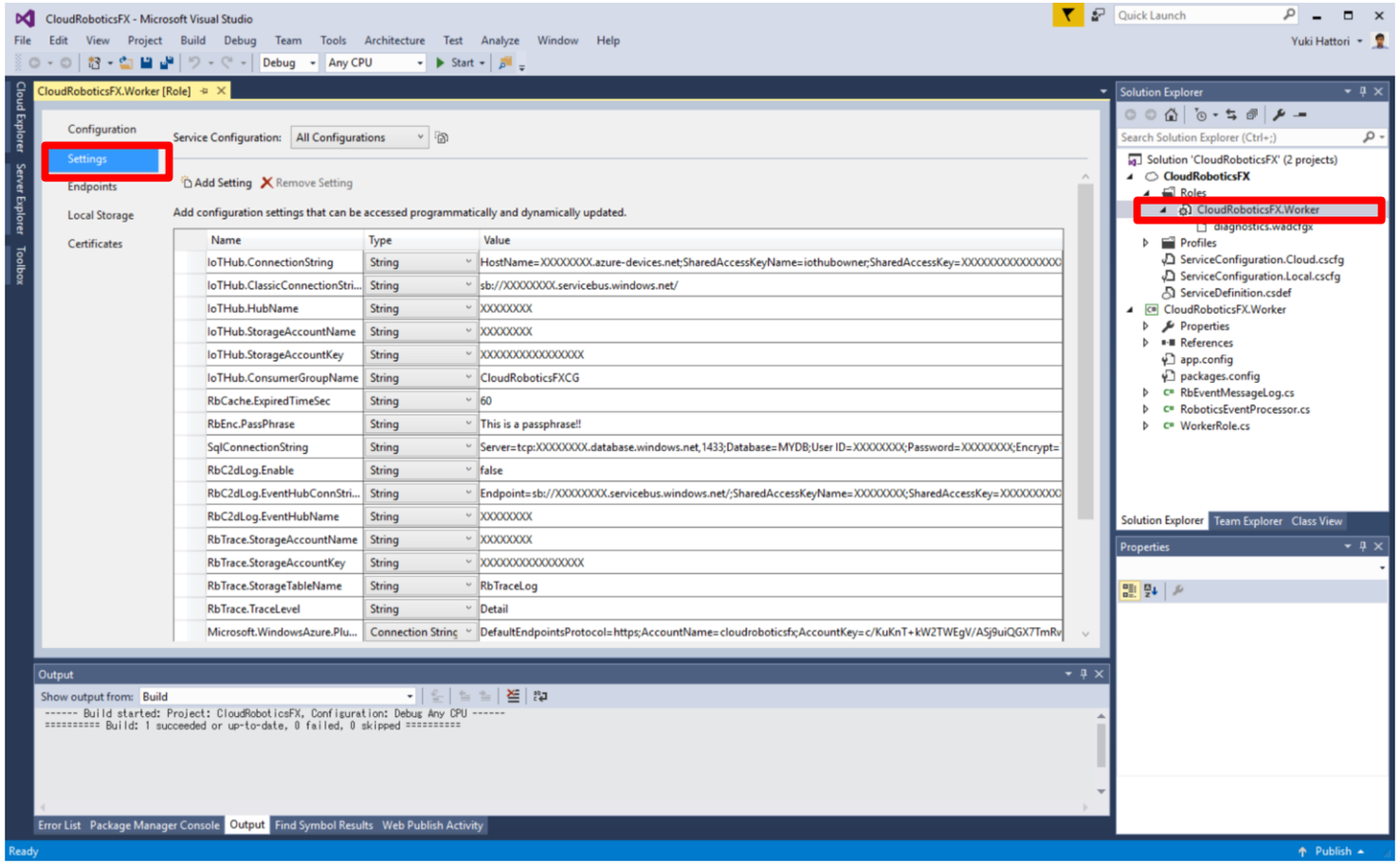
- CloudRoboticsDefTool (Definition & Management Tool)
- CloudRoboticsDeviceSimulator (Device Simulator)
- CloudRoboticsFX (Application Framework)
- CloudRoboticsTestDriver (Test Driver)



T05_03 : Setting up Cloud Robotics FX

Configure Cloud Robotics FX (App Framework / Cloud Service)

- Open the "CRSDKv1¥CloudRoboticsFX" folder and double click "CloudRoboticsFX.sln"
- Double-click "CloudRoboticsFX.Worker" under the "Role" on the "Solution Explorer"
- Click the [Settings] tab and set the parameter value



T05_04： Setting up Cloud Robotics FX

Set the following parameters of Cloud Robotics FX

- IoTHub.ConnectionString (IoT Hub connection string)
  → Set [IoT Hub connection string] saved in Notepad at T02_03
- IoTHub.ClassicConnectionString (Event hub compatible connection string)
  → Set the following SharedAccessKey of [Event hub compatible end point] and [IoT Hub connection string] saved in Notepad
  → Endpoint=sb://xxxxxxxxxx.servicebus.windows.net/;SharedAccessKeyName=iothubowner;SharedAccessKey=xxxxxxxxxxxxxxxxxxxxxxx
  * Since there are many problems, we improved to automatically generate a connection string simply by copying sb: //xxxxxxxxxx.servicebus.windows.net/
- IoTHub.HubName (Event hub compatible name)
  → Set [event hub compatible name] saved in Notepad with T02_04

- IoTHub.StorageAccountName (Storage account name for Read Pointer) : Set [Storage Account Name] saved in Notepad at T01_05
- IoTHub.StorageAccountKey (Storage key for Read Pointer) : Set [key1] saved in Notepad at T01_05
- IoTHub.ConsumerGroupName (IoT Hub Consumer group name) : Set "CloudRoboticsFXCG"

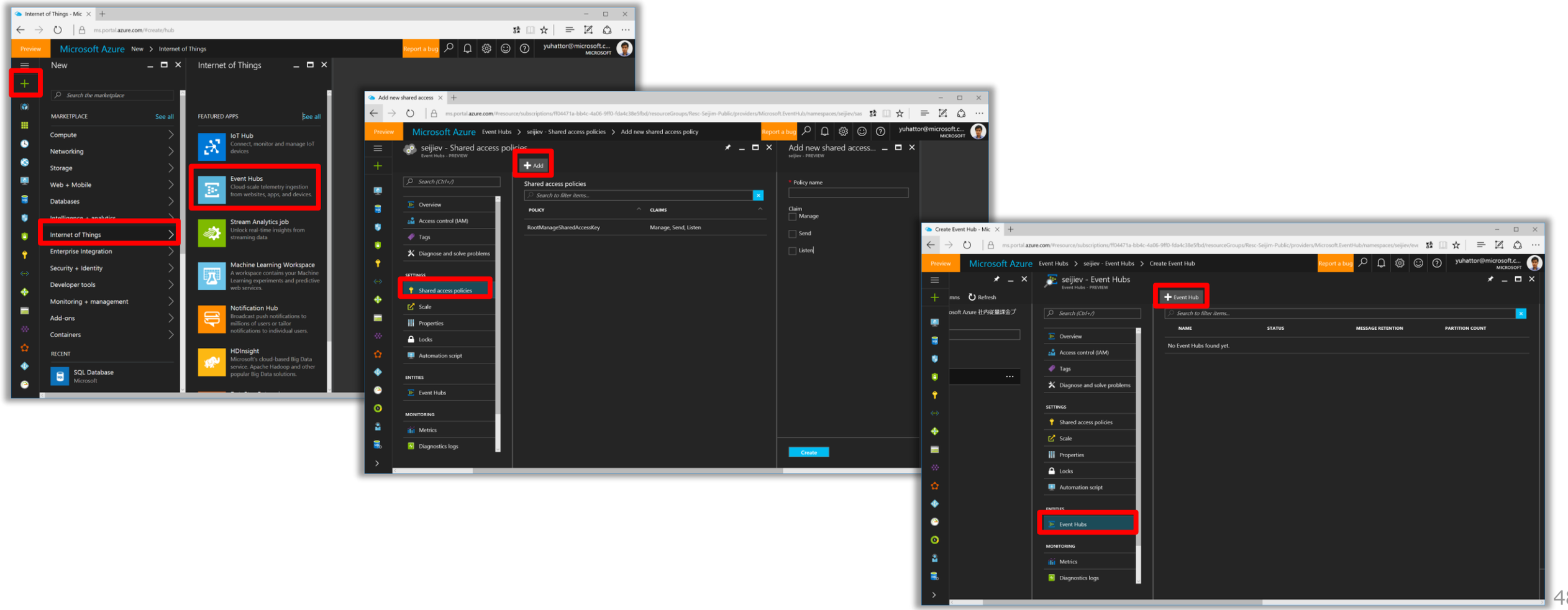
- RbCache.ExpiredTimeSec (Cache seconds for RBFX.AppMaster, AppRouting) : Set "60"
- RbEnc.PassPhrase (Encrypted passphrase for items such as RBFX.AppMaster) : <Any encryption passphrase>
- SqlConnectionString (Connection string for SQL Database) : Set the [ADO.NET (SQL authentication)] connection string saved in Notepad at T03_05
- RbC2dLog.Enabled (option) : Enable logging of sent messages from Cloud Robotics FX ("true" or "false")
- RbC2dLog.EventHubConnString (option) : Connection string of Event Hubs
- RbC2dLog.EventHubName (option) : Entity name of Event Hubs

- RbTrace.StorageAccountName (Storage account name for FX trace log) : Set [Storage Account Name] saved in Notepad at T01_05
- RbTrace.StorageAccountKey (Storage Key for FX trace log) : Set [key 1] saved in Notepad at T 01_05
- RbTrace.StorageTableName (Table name for FX trace log) : Set "RbTraceLog"
- RbTrace.TraceLevel (Output mode for FX trace log) : "Detail" (log output for each message reading) or not set

T05_05 ： Setting up Event Hubs (Option)

If you set RbC2dLog.Enabled = "true" with Cloud Robotics FX parameters, configure Event Hubs

1. Create Event Hubs
2. Create Shared access policies (for send only)
3. Create Entities



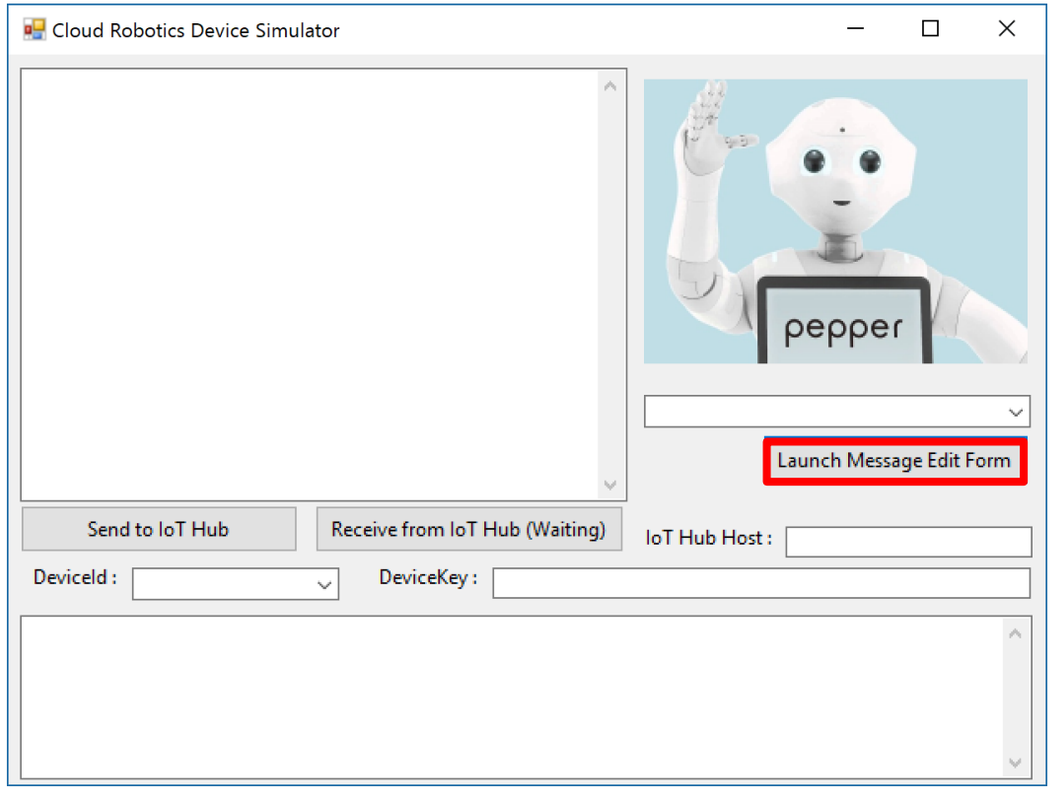
Tutorial 06

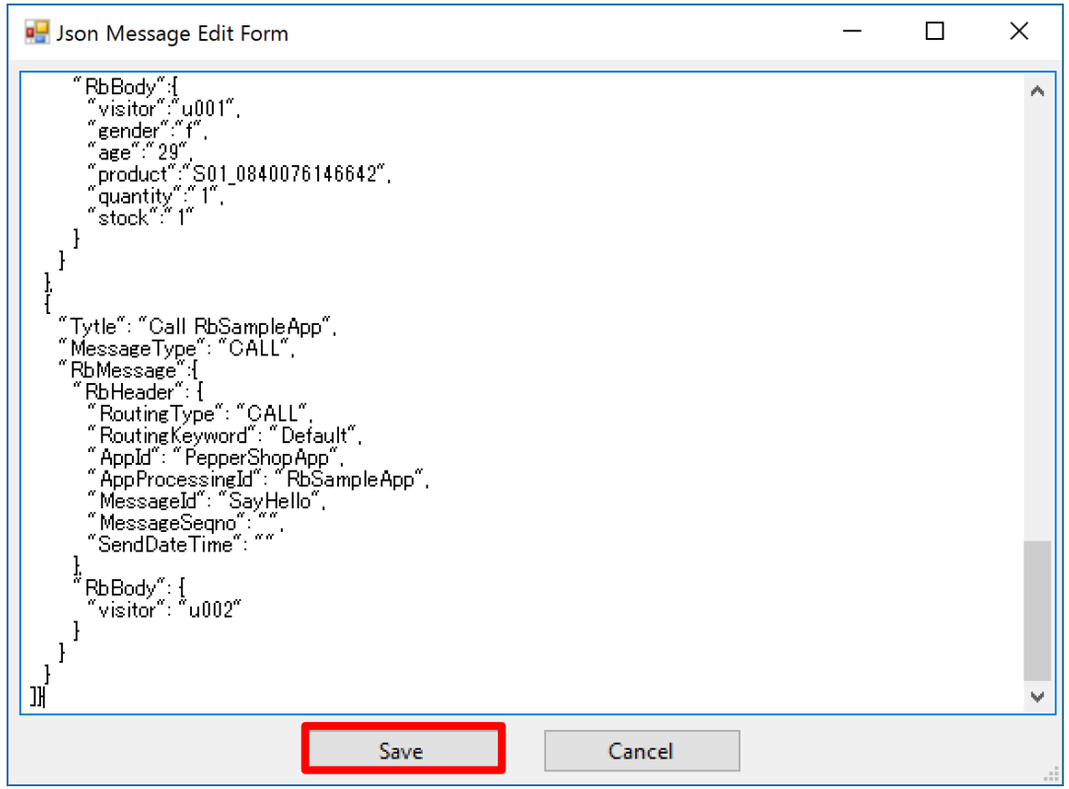
Setting for device-to-device communication

T06_01 : Initial setting of Device Simulator

Configure the communication message template of CloudRoboticsDeviceSimulator

- Go to "CRSDKv1¥CloudRoboticsDeviceSimulator¥CloudRoboticsDeviceSimulator¥bin¥Debug" folder
- Double click "CloudRoboticsDeviceSimulator.exe"
- Press the "Launch Message Edit Form" button to open the communication message template edit form
- Open "CRSDKv1¥CloudRoboticsDeviceSimulator¥DeviceSimulatorMessages.txt" with a notepad etc, copy it and save it

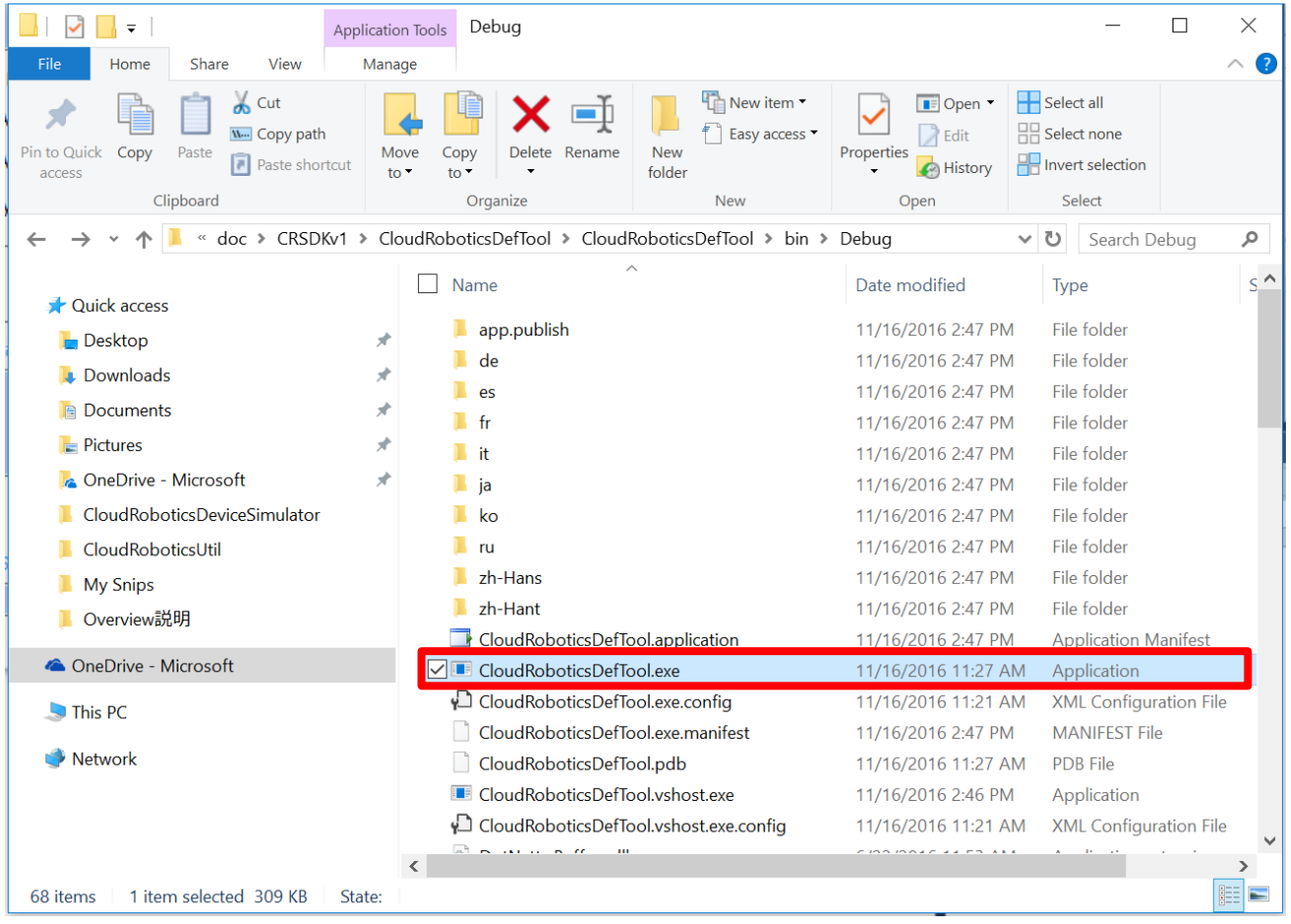




T06_02：Create a shortcut to Cloud Robotics Definition Tool

Create a shortcut to the Cloud Robotics Definition Tool (Management Tool) built on T05_02

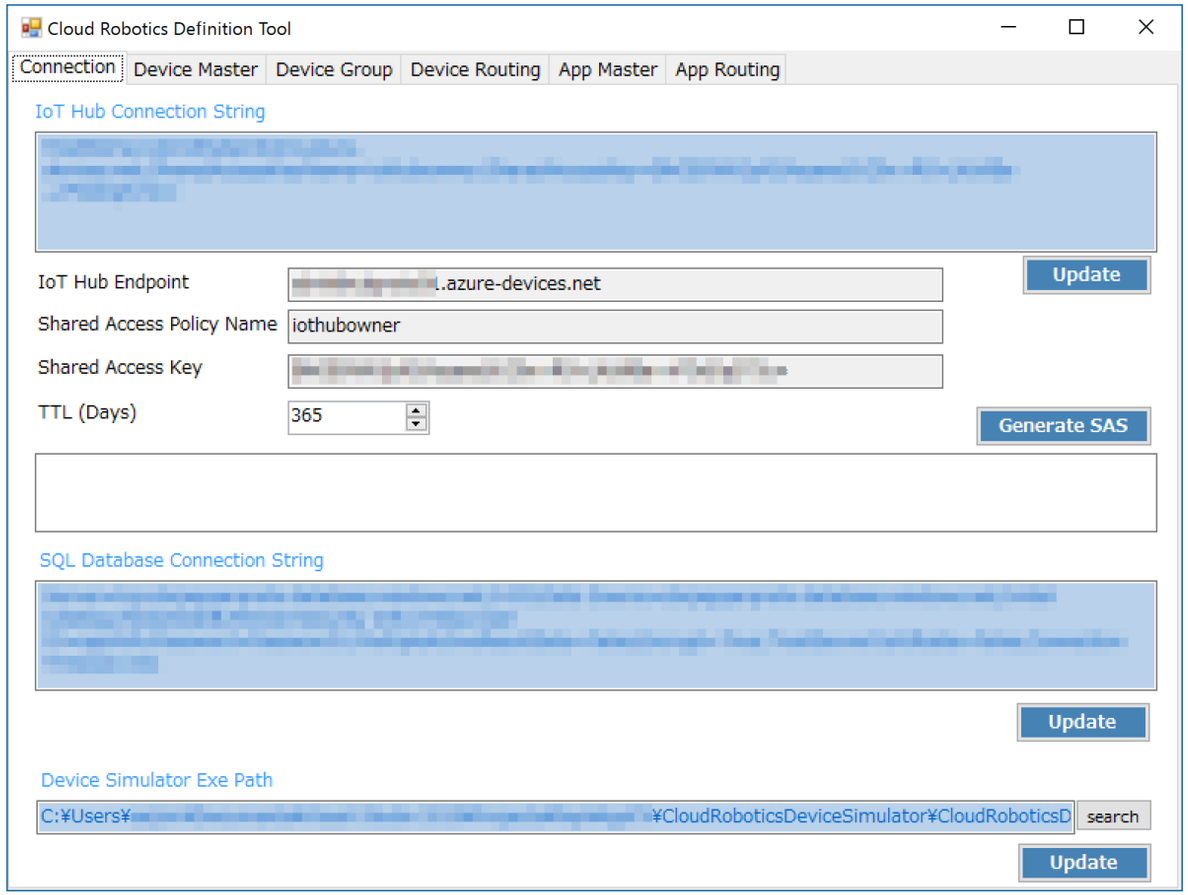
- Go to "CRSDKv1¥CloudRoboticsDefTool¥CloudRoboticsDefTool¥bin¥Debug"
- Please create a shortcut for "CloudRoboticsDefTool.exe" on the desktop etc.



T06_03 : Initial setting of Cloud Robotics Definition Tool

Launch the Cloud Robotics Definition Tool (Management Tool) from the shortcut, set the following, and then [Update]

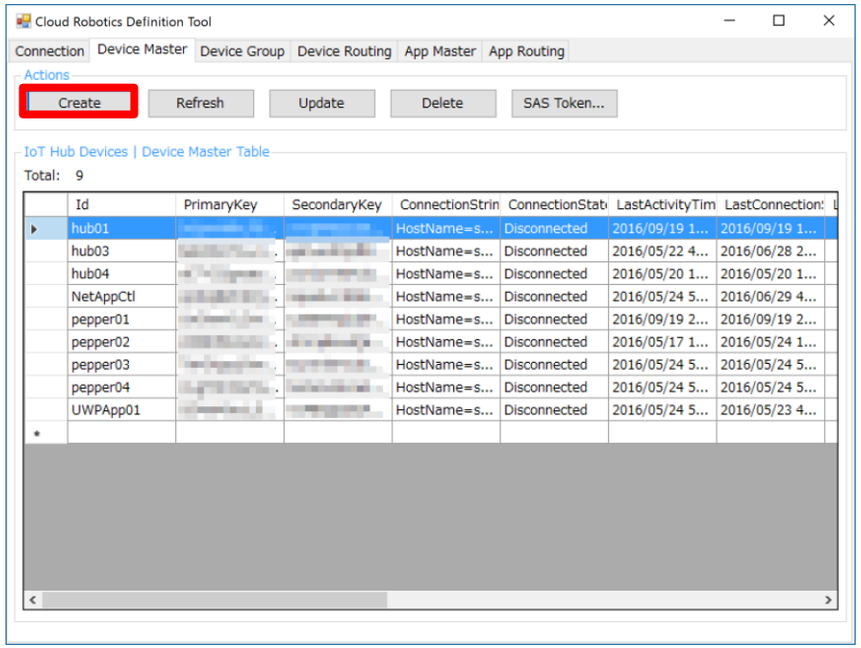
- [IoT Hub Connection String]: Set [IoT Hub connection string] saved in Notepad at T02_03
- [SQL Database Connection String]: Set the [ADO.NET (SQL authentication)] connection string saved in Notepad at T03_05
- Device Simulator Exe Path]: Set the path to CloudRoboticsDeviceSimulator.exe built on T05_02
  → "CRSDKv1¥CloudRoboticsDeviceSimulator¥CloudRoboticsDeviceSimulator¥bin¥Debug¥CloudRoboticsDeviceSimulator.exe"

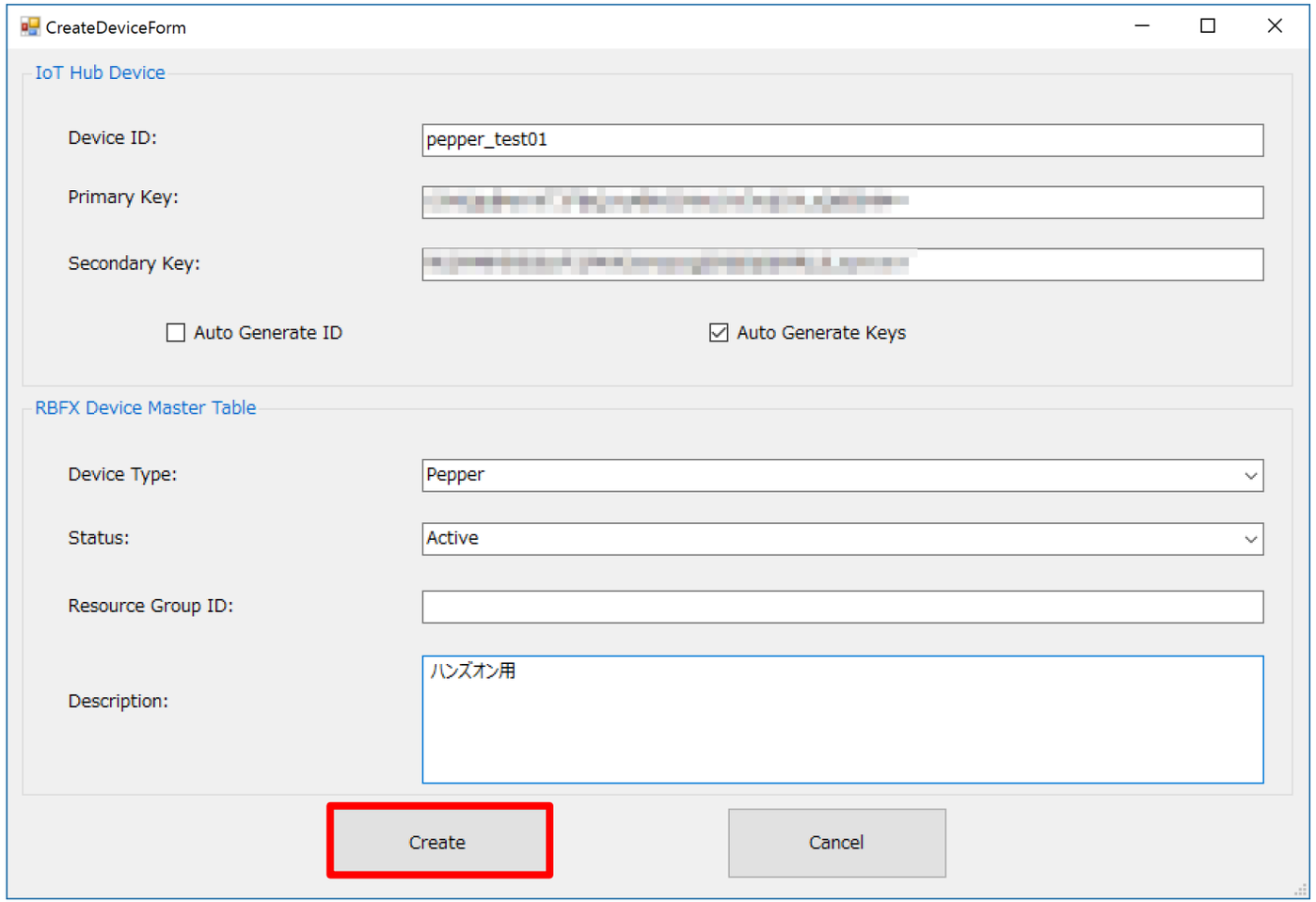


T06_04 : Device registration to IoT Hub & Device Master

Register the following devices from [Device Master] tab → [Create] of Cloud Robotics Definition Tool (Management Tool)

- [Device Id] : "pepper_test01" (Device for Pepper)
- [Device Type] : Select "Pepper"
- [Status] : Select "Active"

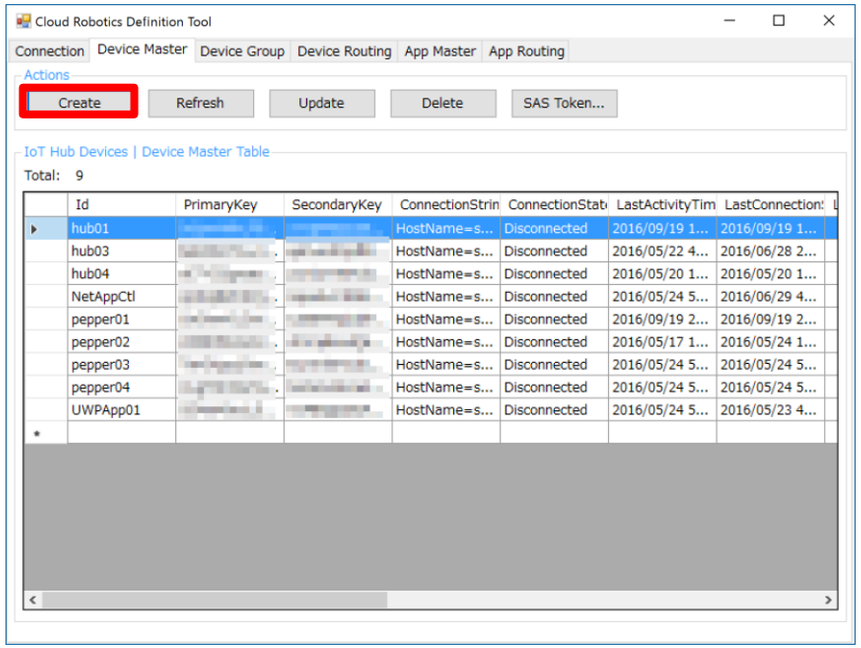


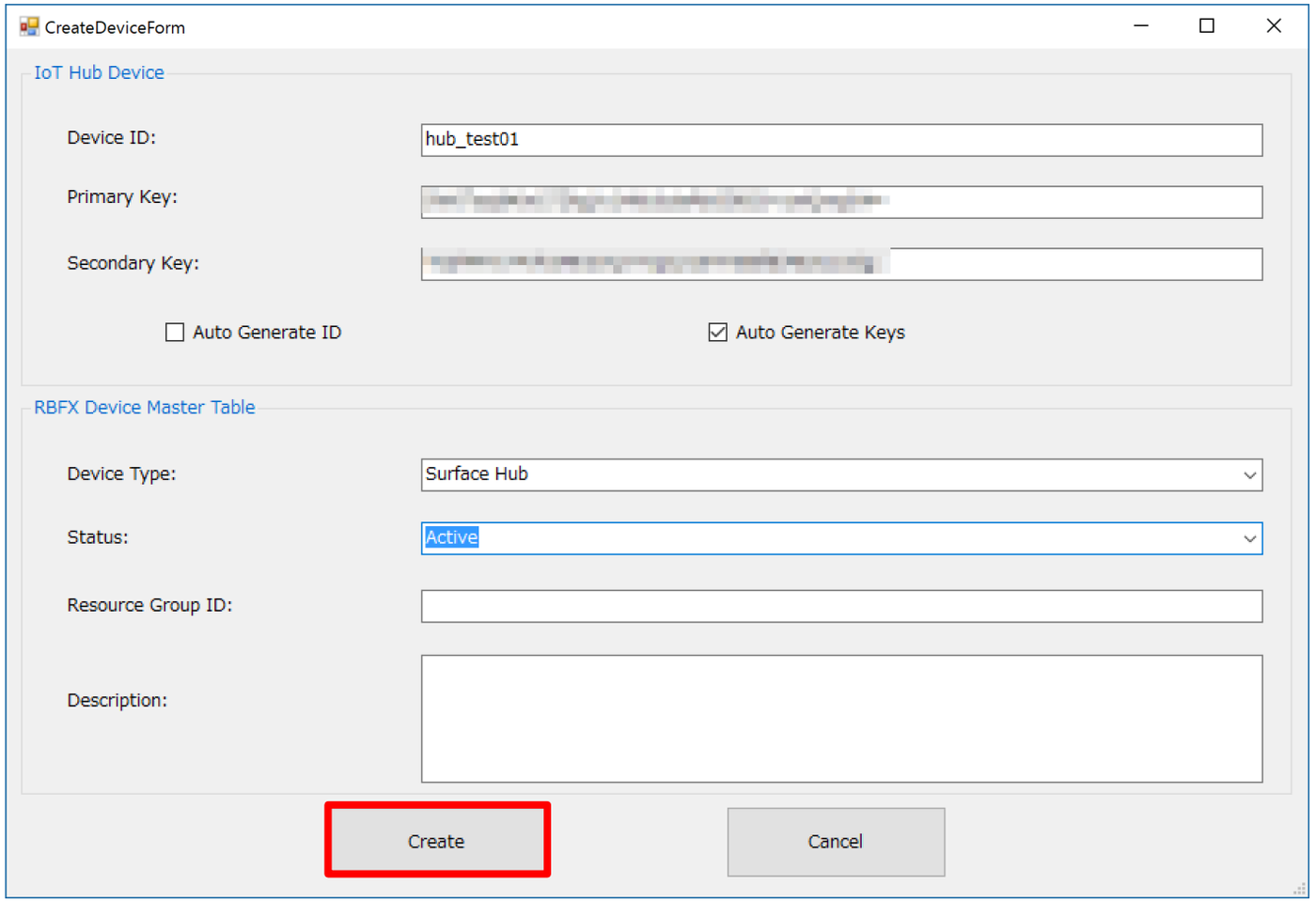


T06_05 : Device registration to IoT Hub & Device Master

Register the following devices from [Device Master] tab → [Create] of Cloud Robotics Definition Tool (Management Tool)

- [Device Id] : "hub_test01" (device for surface hub)
- [Device Type] : Select "Surface Hub"
- [Status] : Select "Active"

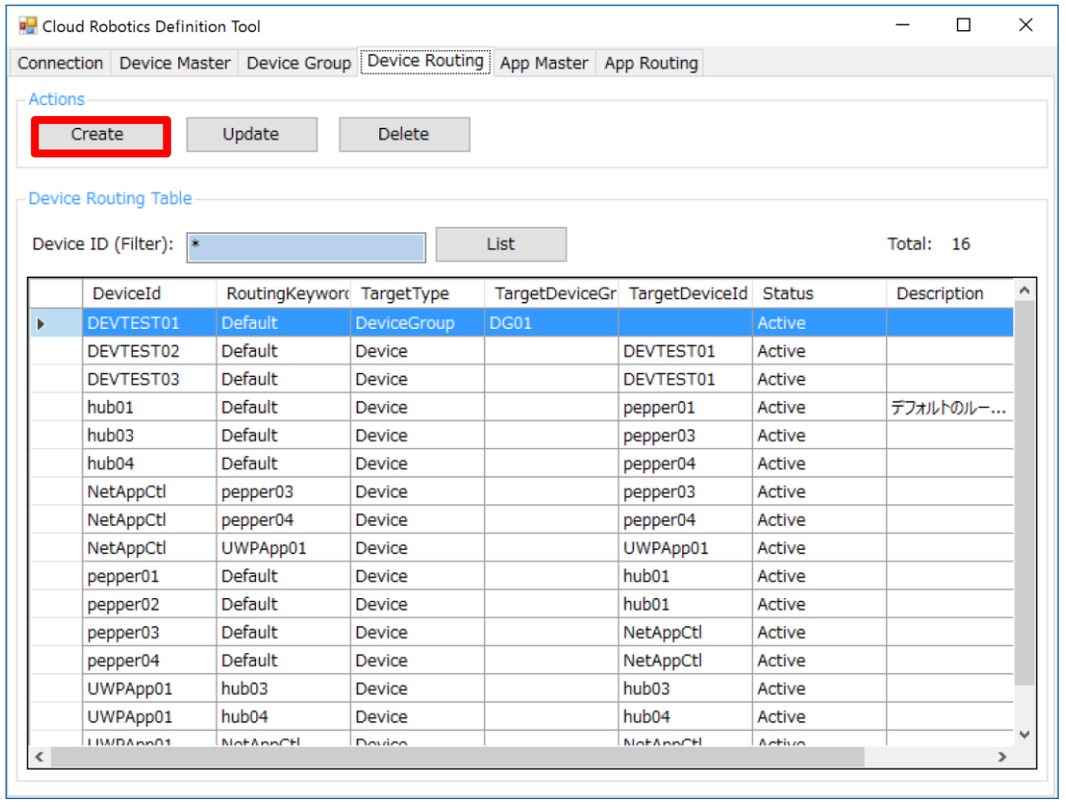


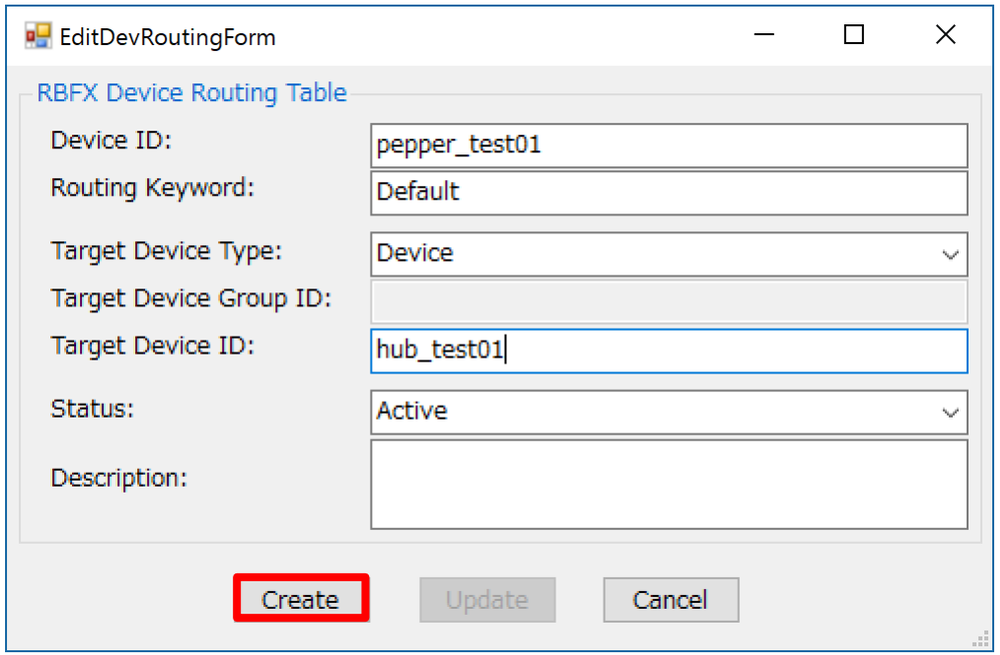


T06_06 : Routing registration to Device Routing

Register the following routing from [Device Routing] tab → [Create] of Cloud Robotics Definition Tool (Management Tool)

- [Device ID] : Enter "pepper_test01"
- [Target Device Type] : Select "Device"
- [Target Device ID] : Enter "hub_test01"

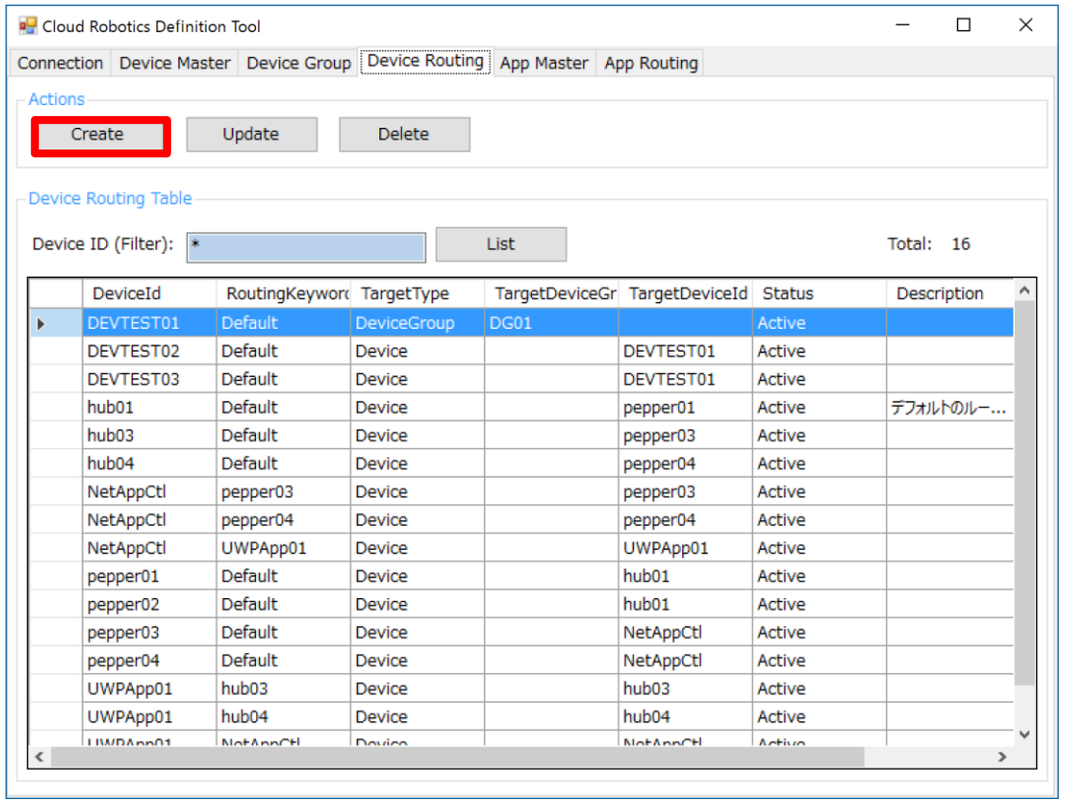


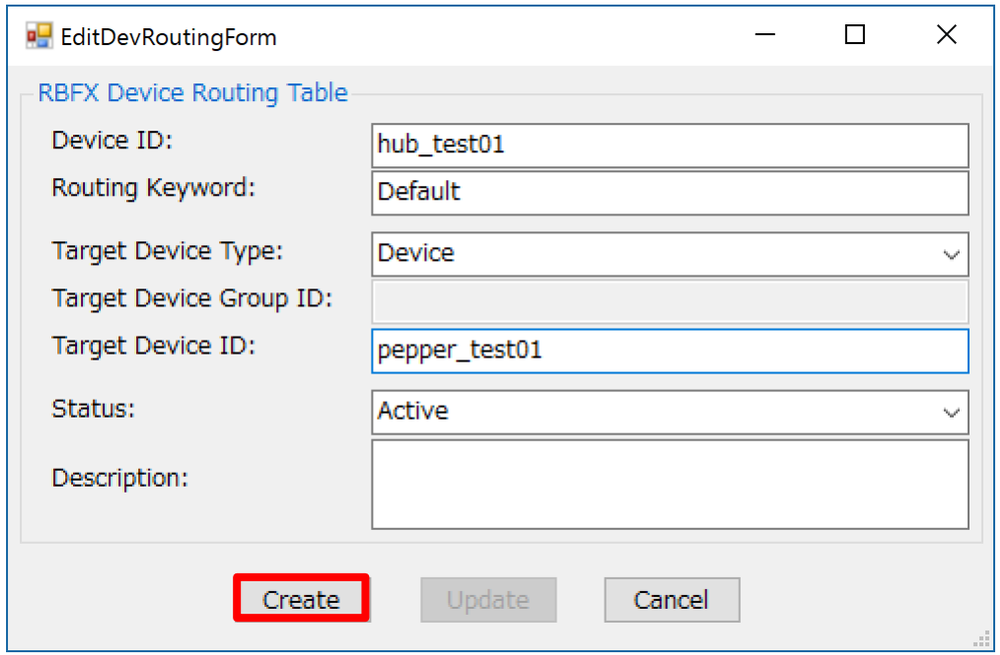


T06_07 : Routing registration to Device Routing

Register the following routing from [Device Routing] tab → [Create] of Cloud Robotics Definition Tool (Management Tool)

- [Device ID] : Enter "hub_test01"
- [Target Device Type] : Select "Device"
- [Target Device ID] : Enter "pepper_test01"





Tutorial 07

Testing communication between devices (Debug execution of Cloud Robotics FX)

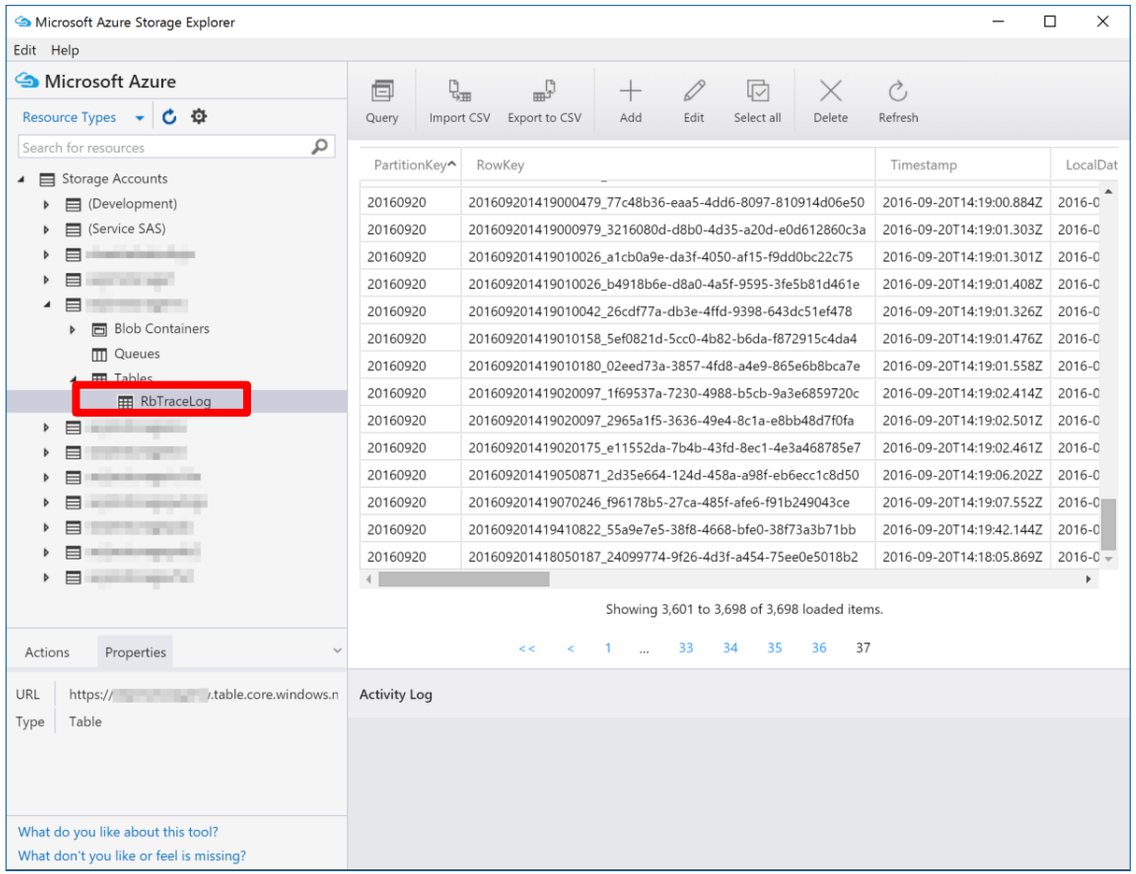
Important

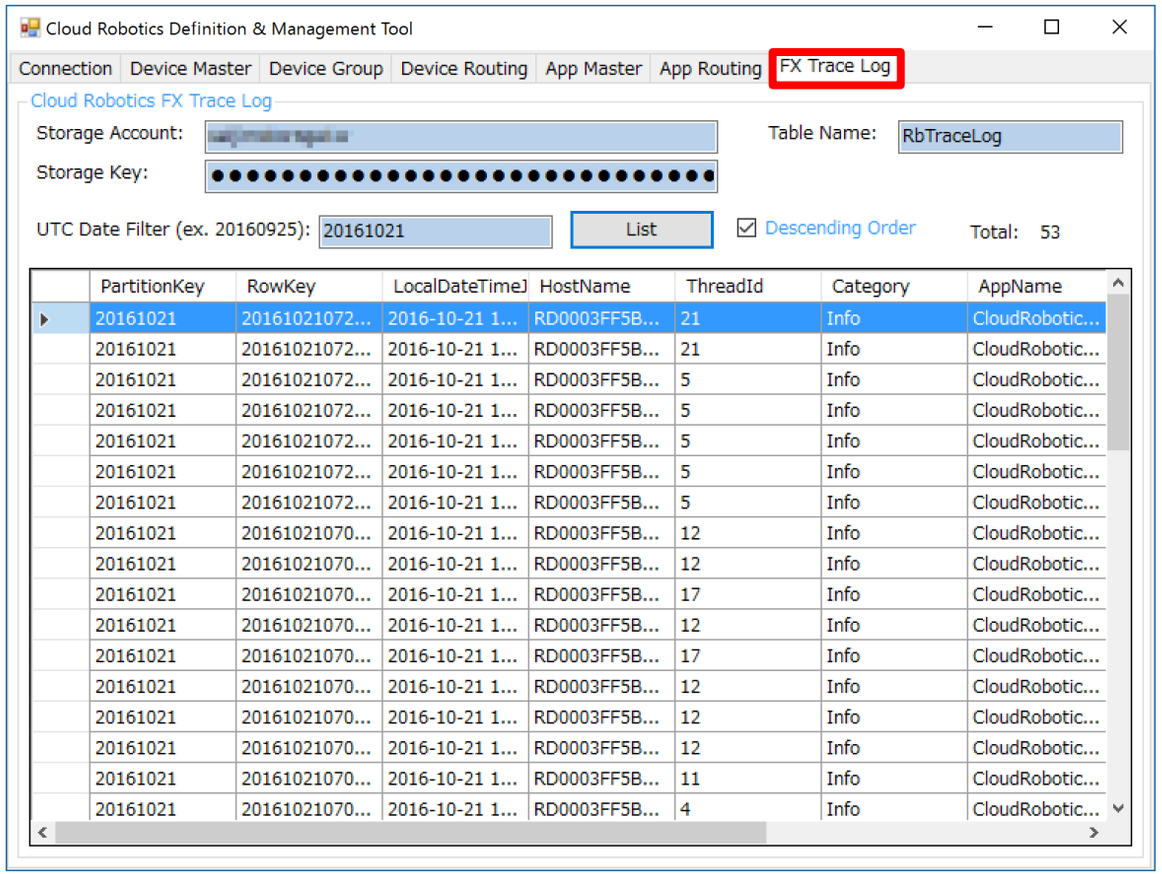
Confirm Cloud Robotics FX trace log

Use Microsoft Azure Storage Explorer that you had installed, or use the Cloud Robotics Definition Tool

- Log in like the Azure portal
- Open the Cloud Robotics FX storage account for tracing log which you set up in T05_04
- Double-click "RbTraceLog" under [Tables] and confirm that there are no errors in the message

* "RbTraceLog" is created when Cloud Robotics FX is started for the first time



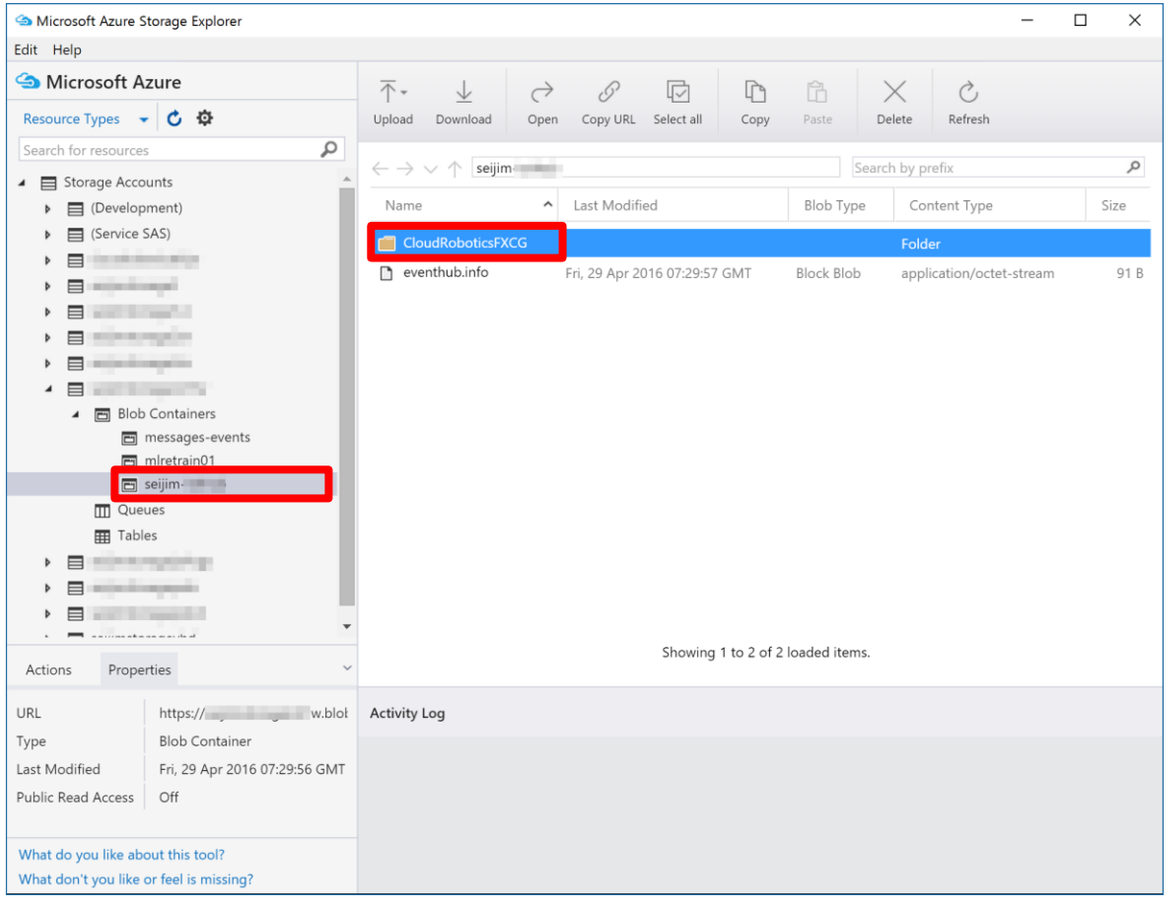


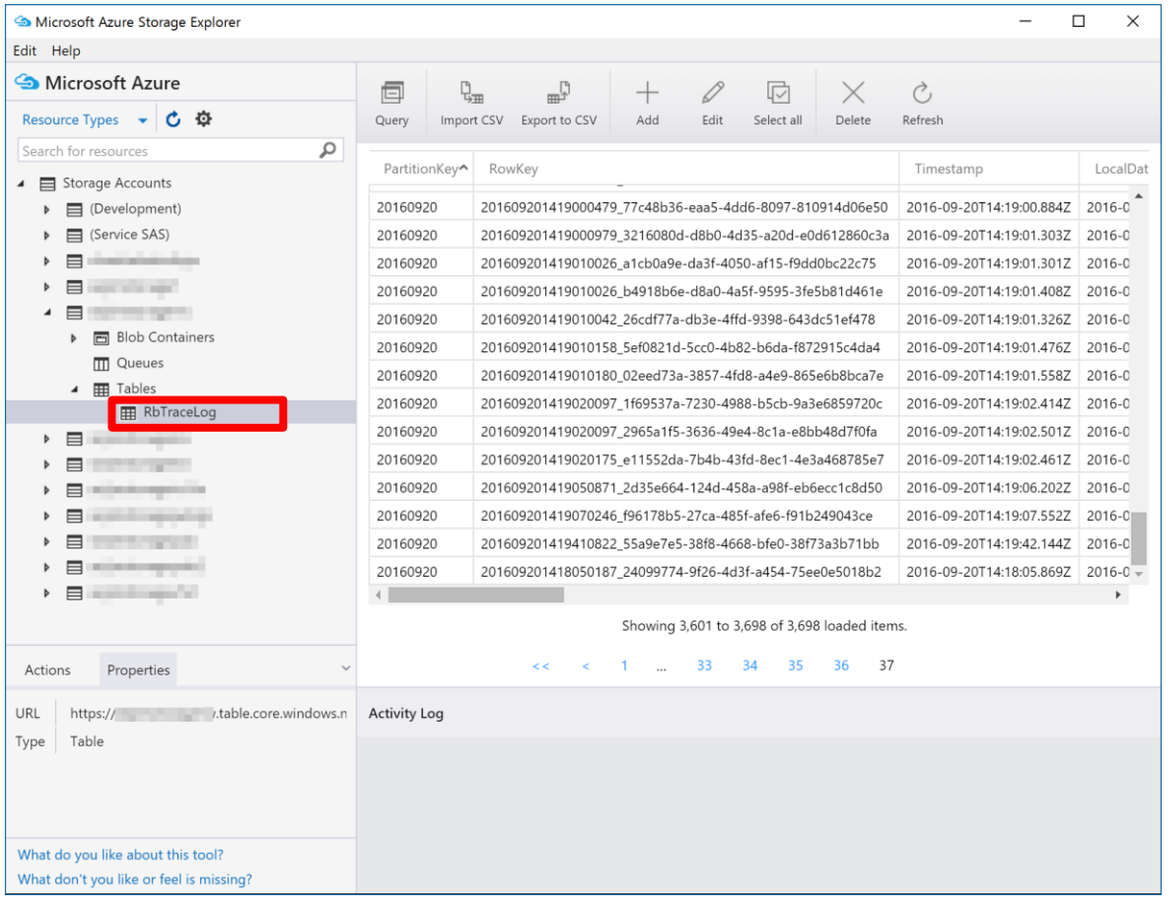
Important

Troubleshoot Cloud Robotics FX

When debugging is executed, when the cursor does not come to the break point or when inter-device communication does not go well

- Start Microsoft Azure Storage Explorer
- Confirm that the BLOB container (<IoT Hub name>) is created in the storage account for Read Pointer set in T05_04
- Confirm that there is no error in Cloud Robotics FX trace log "RbTraceLog"
- Suspected incorrect FX setting parameter in T05_04, mismatch of encrypted passphrase between CloudRoboticsDefTool and FX

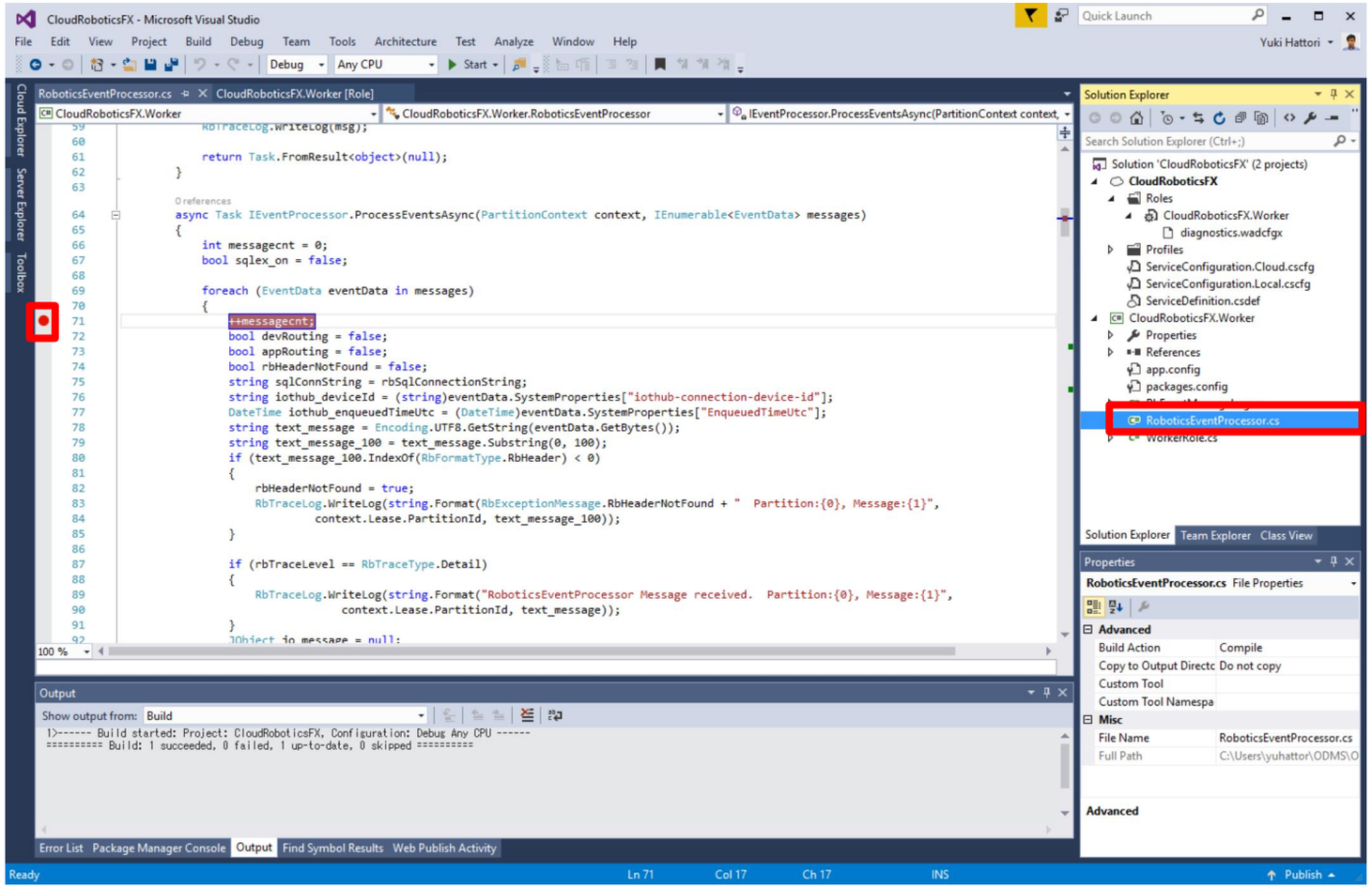




T07_01 : Debug execution of Cloud Robotics FX

Open Cloud Robotics FX solution file

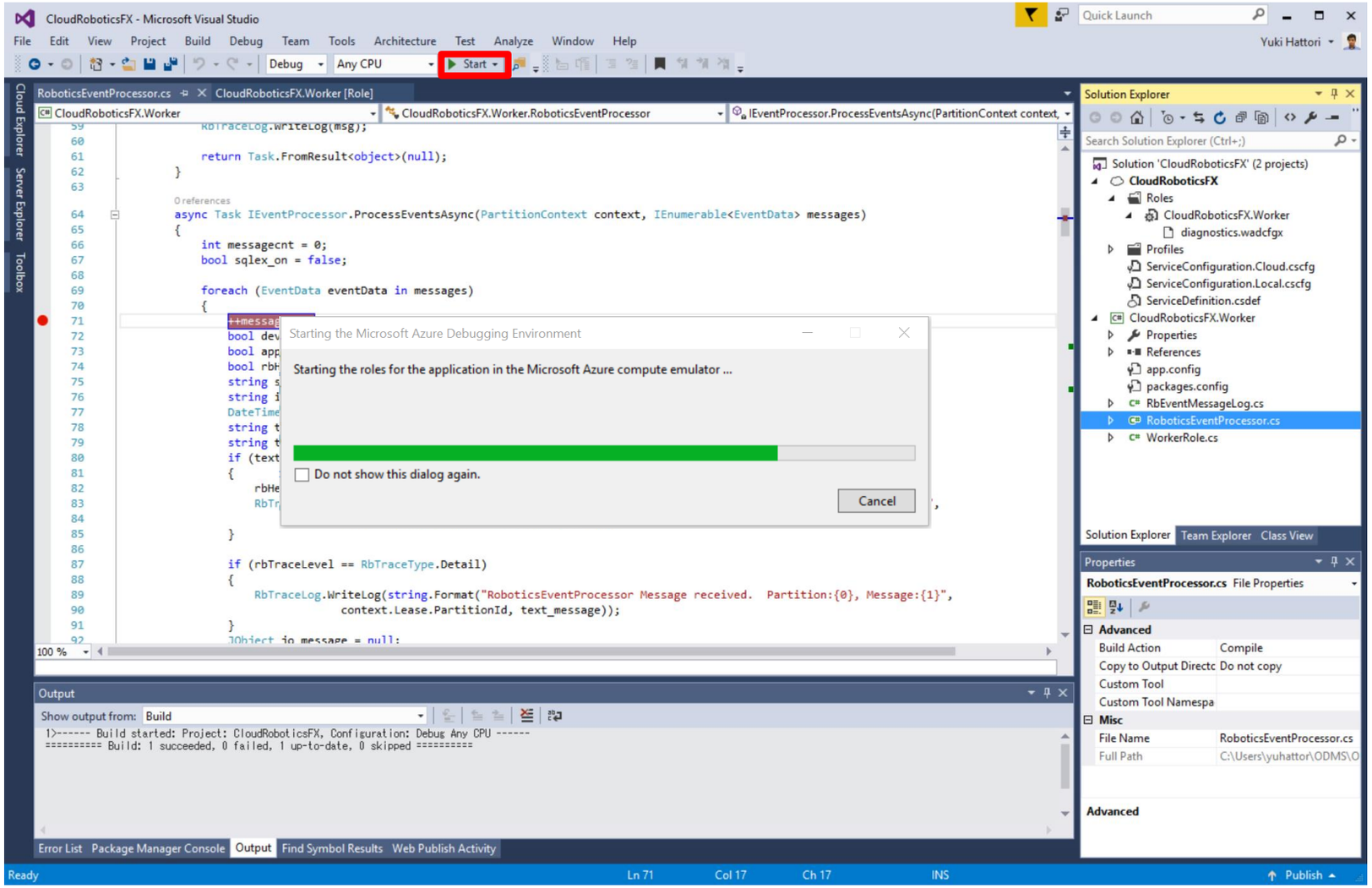
- Go to "CRSDKv1¥CloudRoboticsFX" and double click "CloudRoboticsFX.sln"
- Open "RoboticsEventProcessor.cs" under the "CloudRoboticsFX.Worker" project on Solution Explorer
- Set a breakpoint in the "foreach" statement of "async Task IEventProcessor.ProcessEventsAsync" method



T07_02 : Debug execution of Cloud Robotics FX

Run Cloud Robotics FX

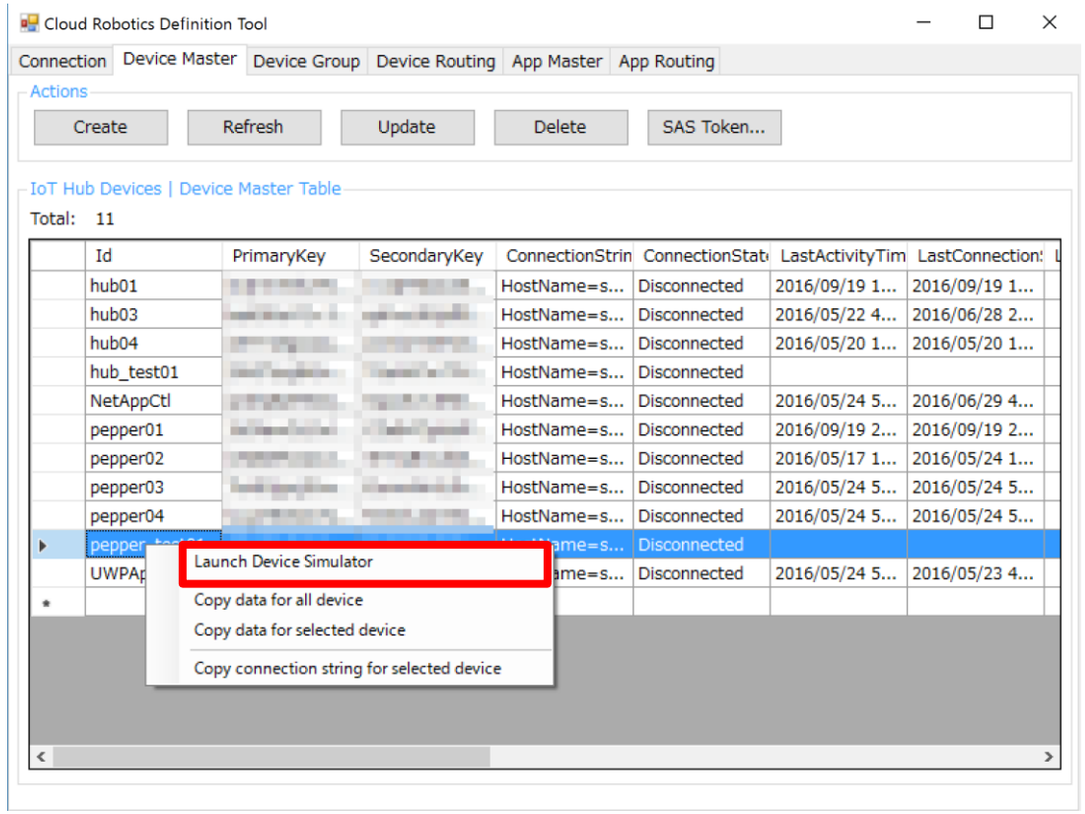
- From the menu bar, press [▶Start]

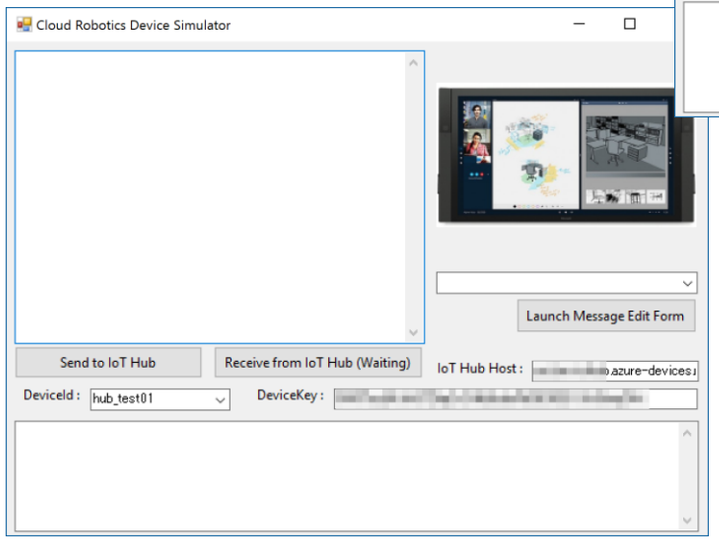


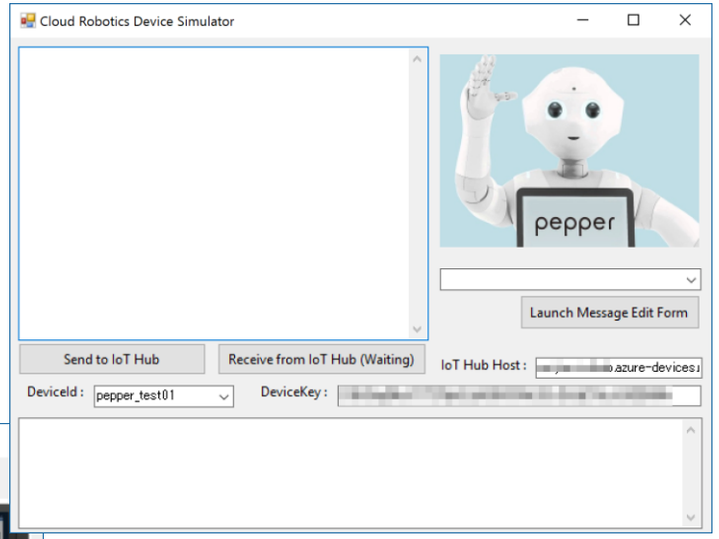
T07_03 : Test with Device Simulator

Launch the Cloud Robotics Definition Tool (Management Tool) from the shortcut

- Go to the Device Master tab
- Select [id] = "pepper_test01", then right click and click "Launch Device Simulator"
- Similarly, select [id] = "hub_test01", then right click and then click "Launch Device Simulator"



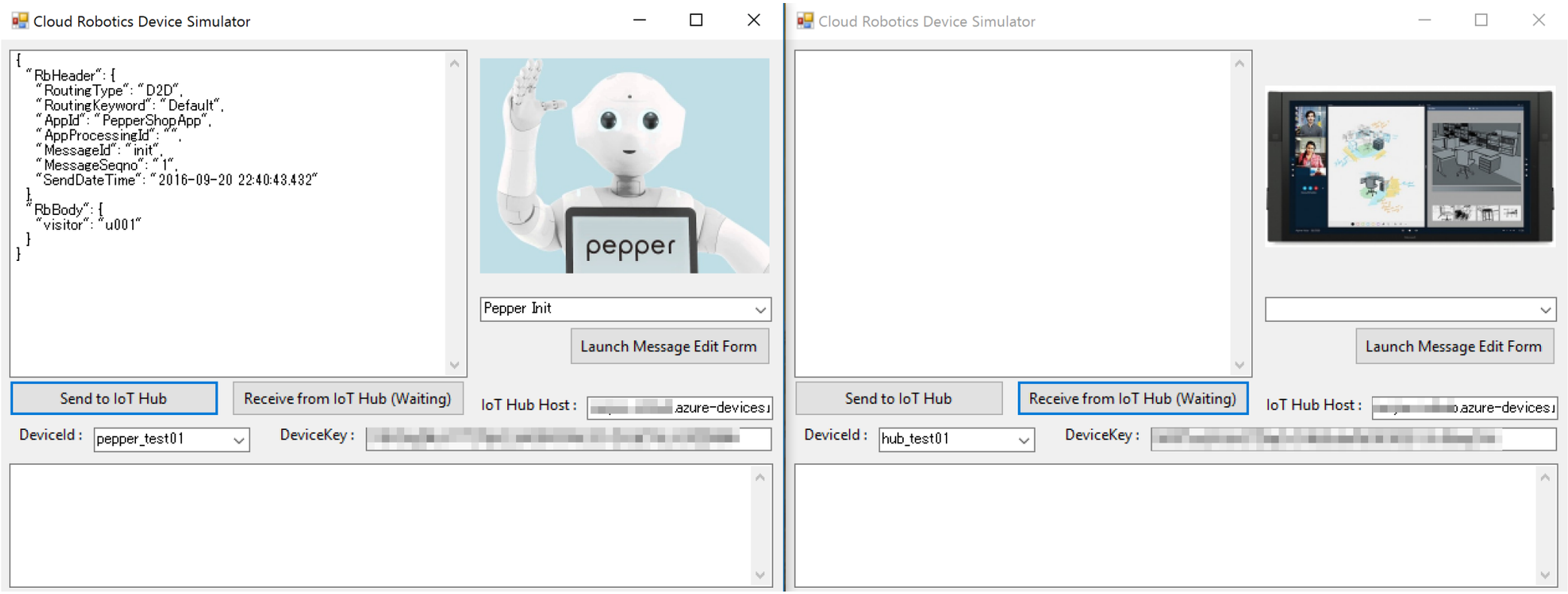




T07_04 : Test with Device Simulator

Execute device routing test using Cloud Robotics Device Simulator

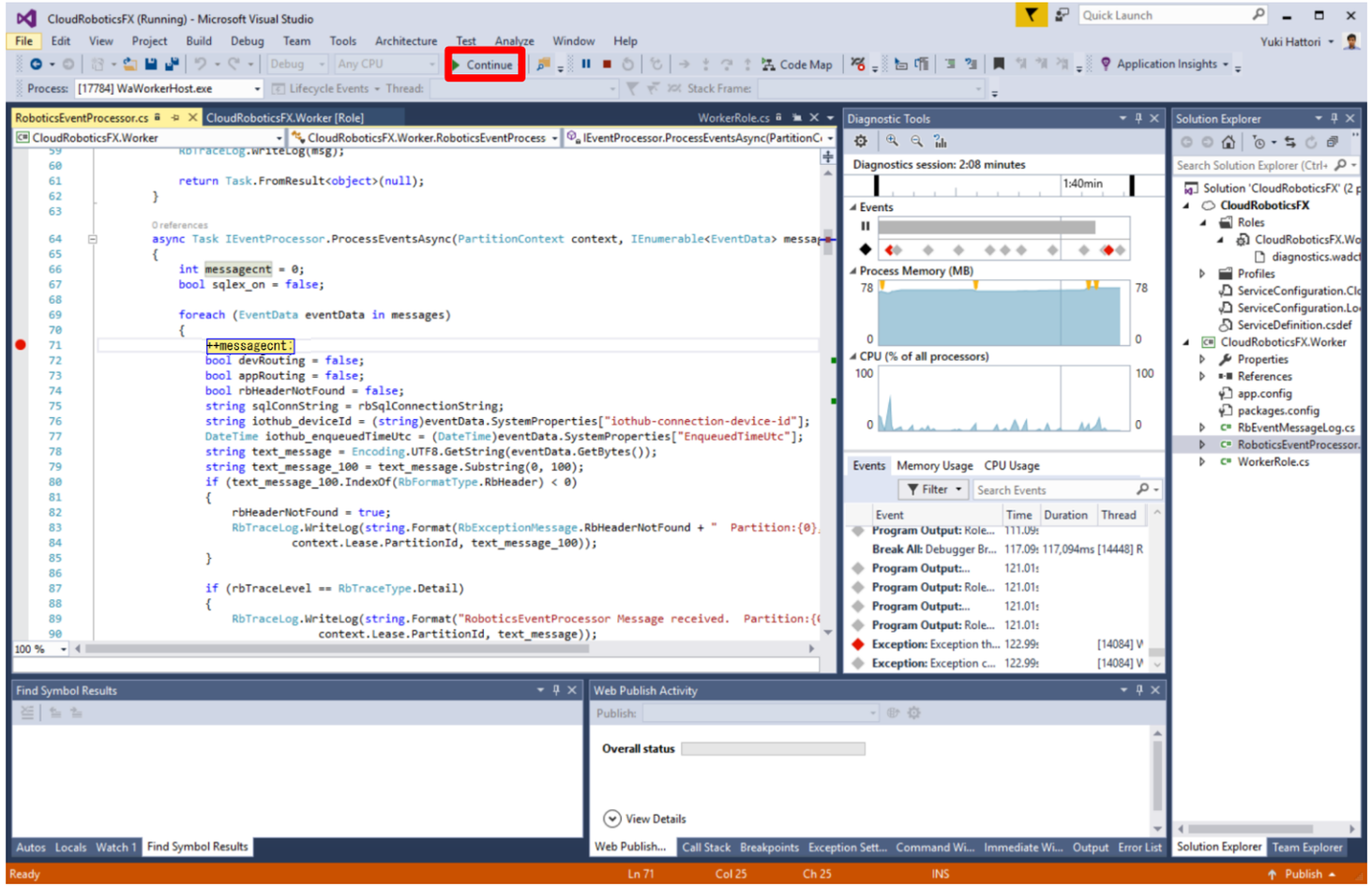
- Press the [Receive from IoT Hub (Wating)] button in the simulator of "hub_test 01" (Surface Hub)
- Press the "Receive from IoT Hub (Wating)" button in the simulator of "pepper_test 01" (Pepper)
- Select "Pepper Init" from the communication message template in the simulator of "pepper_test 01" (Pepper),
- Press the [Send to IoT Hub] button in the simulator of "pepper_test 01" (Pepper)

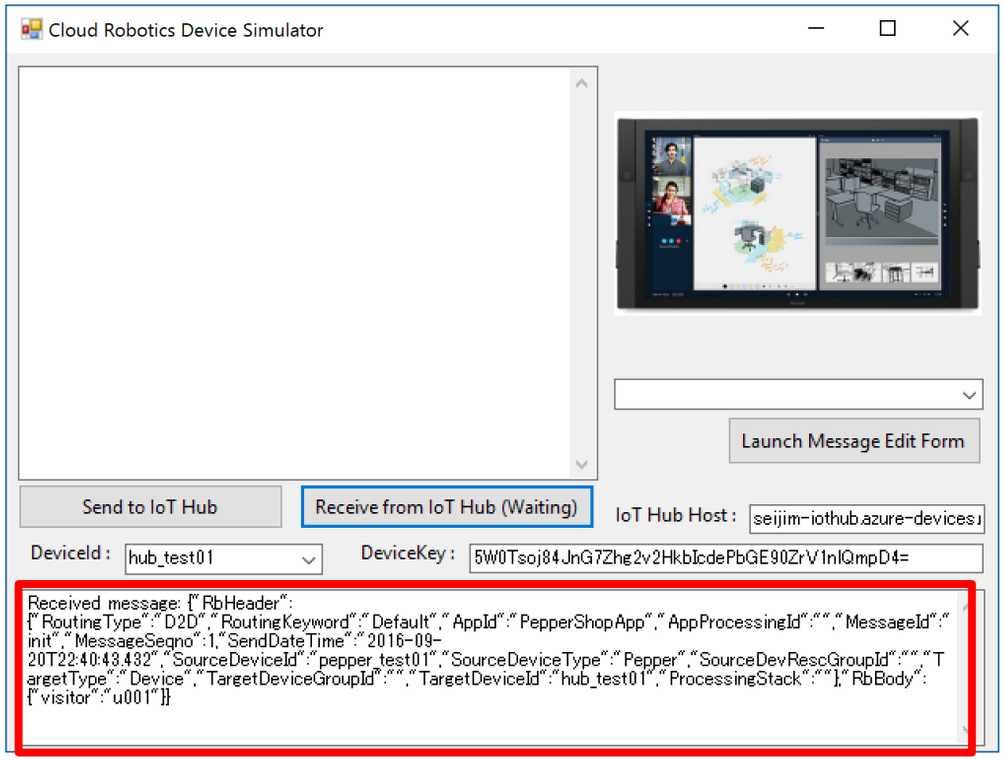


T07_05 : Step execution of Cloud Robotics FX

If the message transmission from Device Simulator was successfully read, the cursor is at the break point

- If the cursor is on the break point, press the [▶ continue] button (or F5)
- If a message has arrived at the Device Simulator of "hub_test01" (Surface Hub), communication between devices is successful





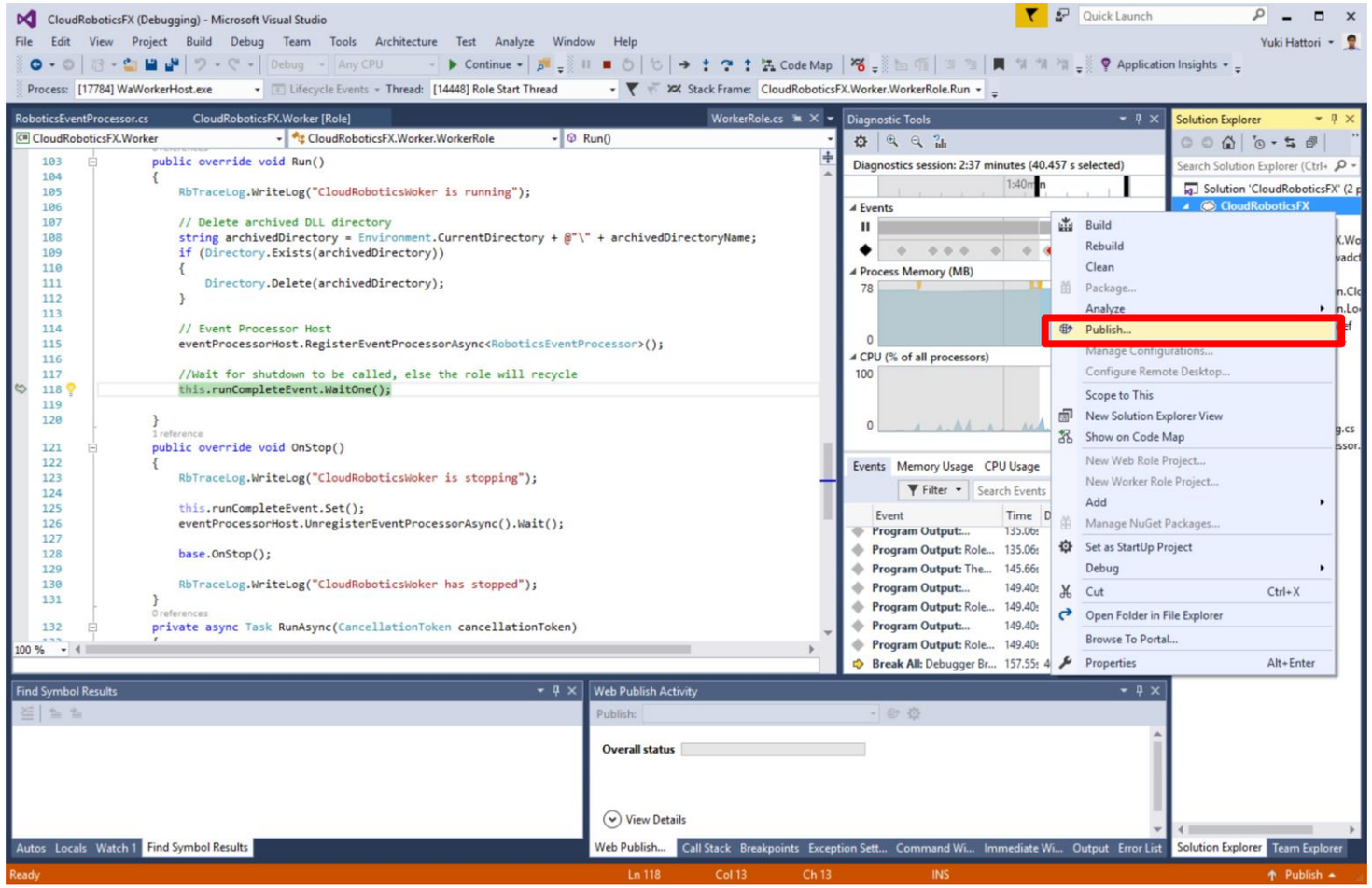
Tutorial 08

Deploying Cloud Robotics FX

T08_01 : Deployment settings for Cloud Robotics FX

Open Cloud Robotics FX solution file

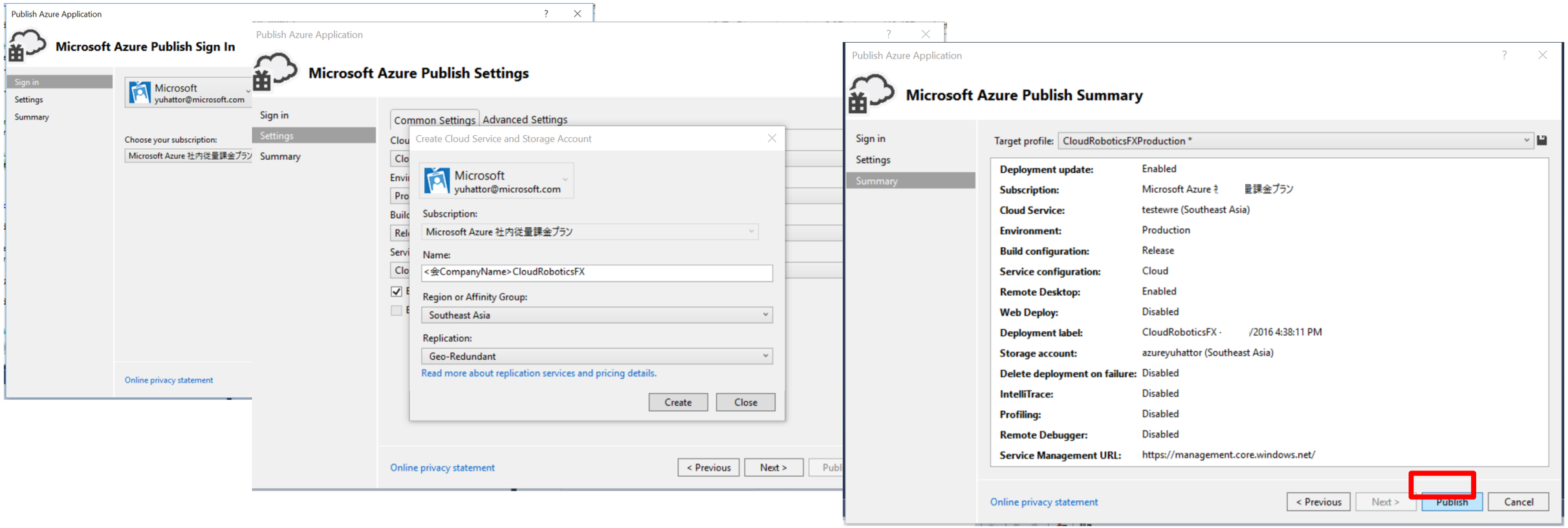
- Double click "CRSDKv1¥CloudRoboticsFX¥CloudRoboticsFX.sln"
- Right-click "CloudRoboticsFX" of Cloudmark icon in Solution Explorer
- Select "Publish"



T08_02：Deployment settings for Cloud Robotics FX

Configure Azure application publishing settings

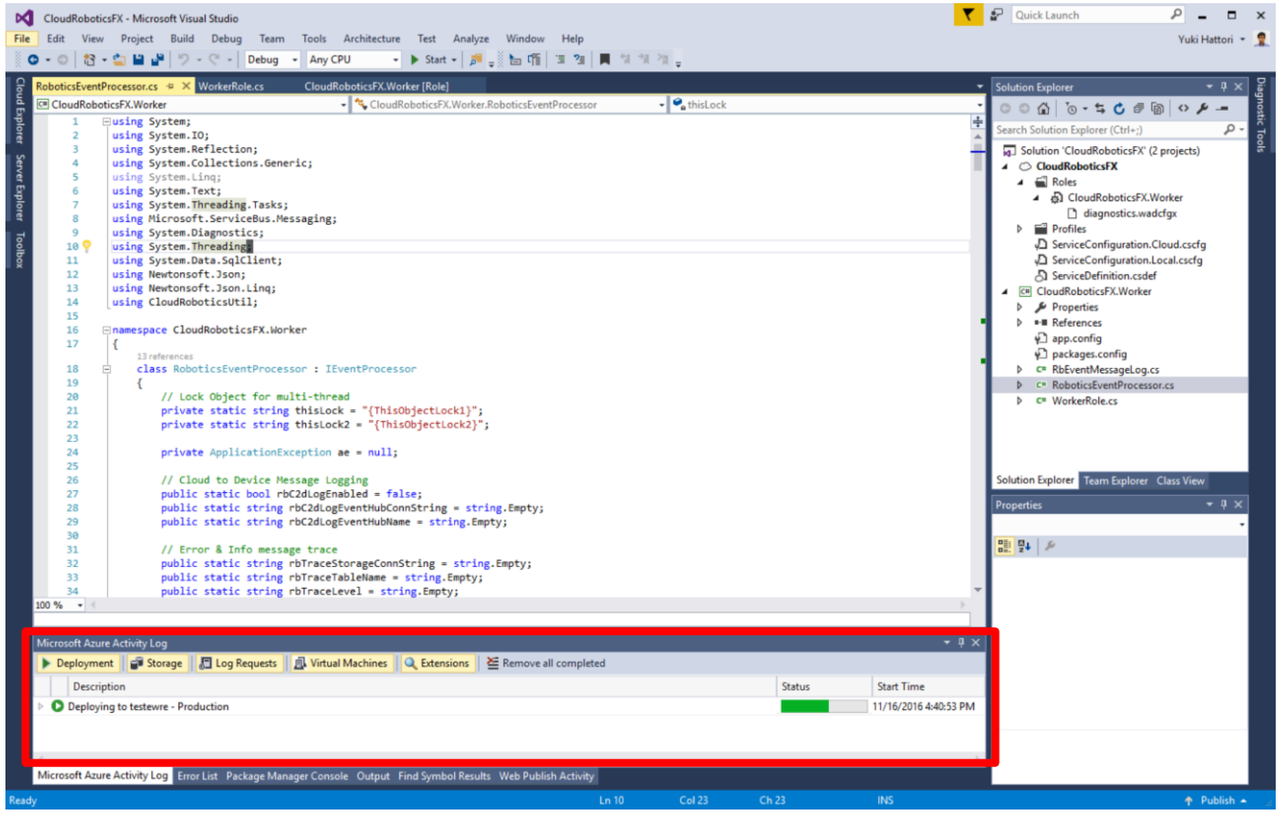
- Sign in to Azure from [Sign In]
- On the [Advanced] tab of [Settings], create a storage with [name] = {<Company name prefix>CloudRoboticsFX>} (ex. SBRCloudRoboticsFX)
- On the [Common Settings] tab of [Settings], select [New] from the [Cloud service] drop-down list
- [Name] = {<Company name prefix> CloudRoboticsFX}, [Region] = {East Japan or West Japan} and press [Create]
- press [Next], check the contents on the [Summary] tab and press the [Publish] button

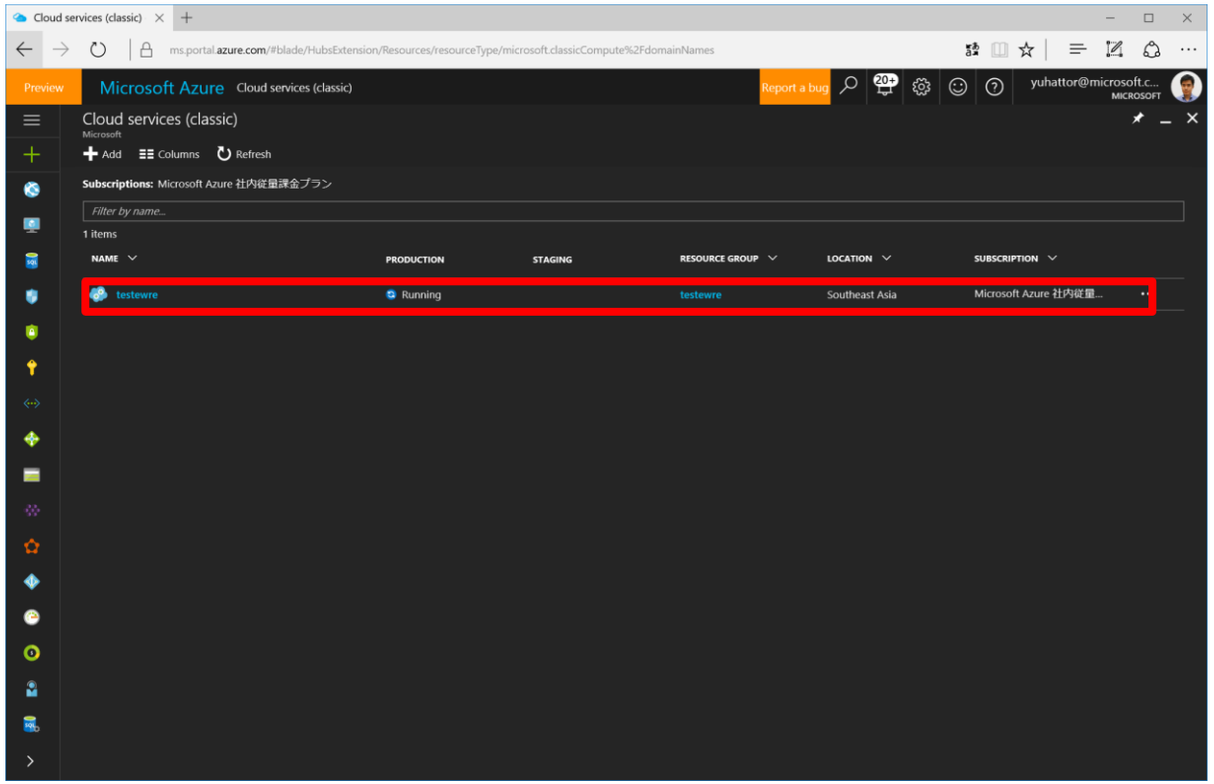


T08_03 : Confirm deployment of Cloud Robotics FX

Configure Azure application publishing settings

- Check the deployment status in Visual Studio. If [status] = {completed}, it is successfully completed.
- From the Azure portal, check the applicable deployment status in [Cloud Services].If [production environment] = {running}, it is successfully completed.

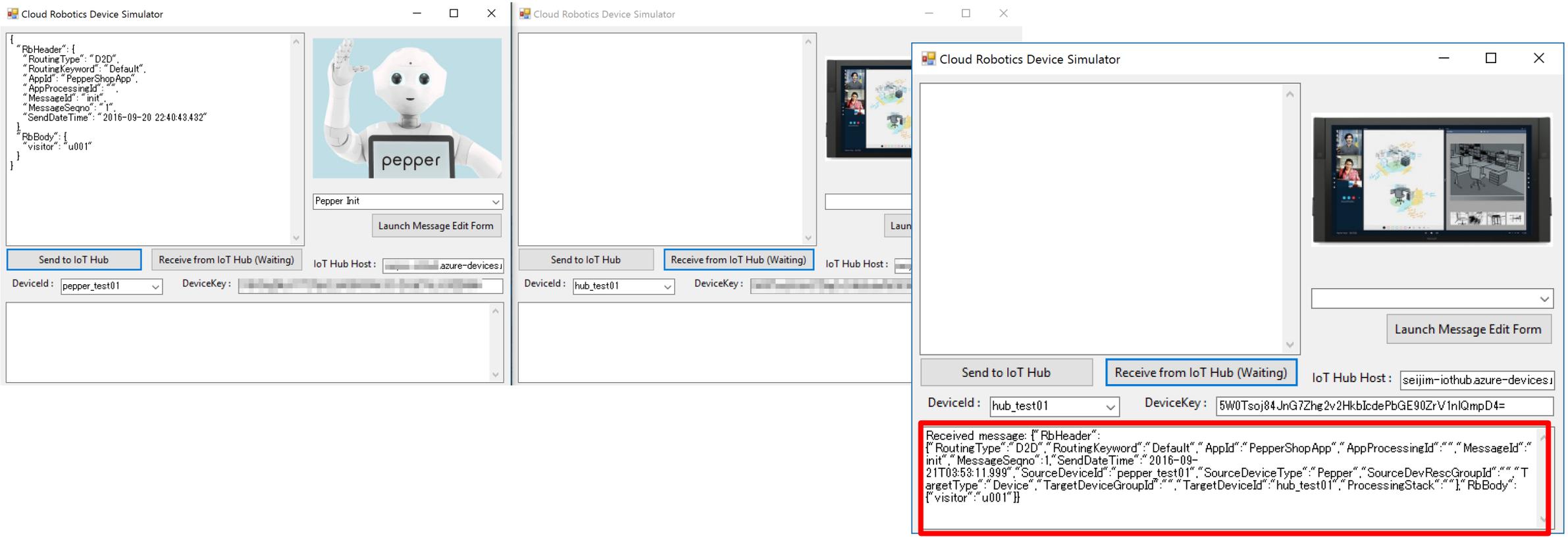




T08_04 : Testing after deployment

Execute device routing test using Cloud Robotics Device Simulator

- Press the [Receive from IoT Hub (Wating)] button in the simulator of "hub_test01" (Surface Hub)
- Press the "Receive from IoT Hub (Wating)" button in the simulator of "pepper_test01" (Pepper)
- Select "Pepper Init" from the communication message template, in the simulator of "pepper_test01" (Pepper)
- Press the [Send to IoT Hub] button in the simulator of "pepper_test01" (Pepper)
- It will be successful if a message is delivered to the simulator of "hub_test01" (Surface Hub)



Tutorial 09

Run the application on Cloud Robotics FX

T09_01：Build a Hello World application (DLL)

Open RbSampleApp solution file

- Open "CRSDKv1¥RbSampleApp¥RbSampleApp.sln" and execute the build

```
using System;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using CloudRoboticsUtil;

namespace RbSampleApp
{
    public class SayHello : MarshalByRefObject, IAppRouterDll
    {
        public JArrayString ProcessMessage(RbAppMasterCache rbappmc, RbAppRouterCache rbapprc, RbHeader rbh, string rbBodyString)
        {
            JArray ja_messages = new JArray();
            AppBody appbody = new AppBody();
            appbody.Hello = "Hello World !!!!!!";

            RbMessage message = new RbMessage();
            message.RbHeader = rbh;
            message.RbBody = appbody;

            string json_message = JsonConvert.SerializeObject(message);
            JObject jo = (JObject)JsonConvert.DeserializeObject(json_message);
            ja_messages.Add(jo);
            JArrayString jaString = new JArrayString(ja_messages);

            return jaString;
        }
    }
}
```

The information of [RBFX].[AppRouter] table is Cached on the FX side (* 1)

The information of [RBFX].[AppMaster] table is Cached on the FX side (* 1)

デバイスへの出力メッセージを複数セット可能
(＝複数のデバイスに異なるメッセージを送信可能)

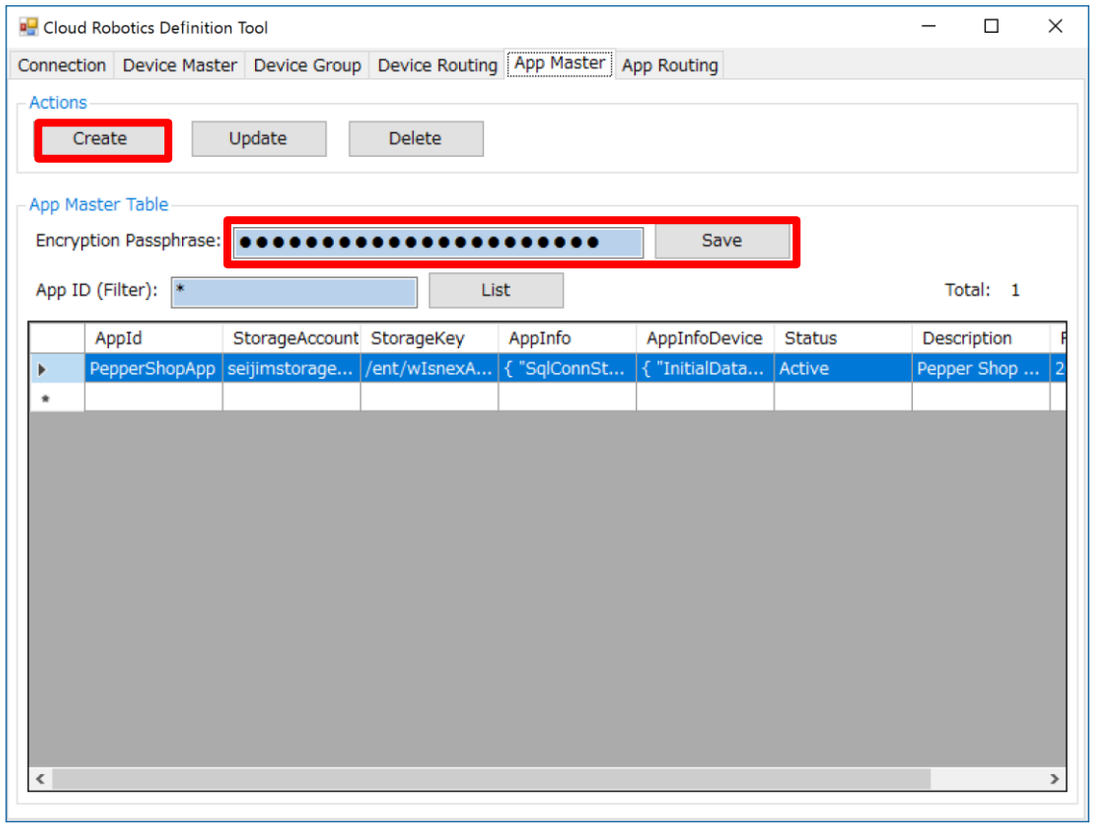
***1 : Set by cscfg file on the FX. The default value of cached time is 60 seconds**

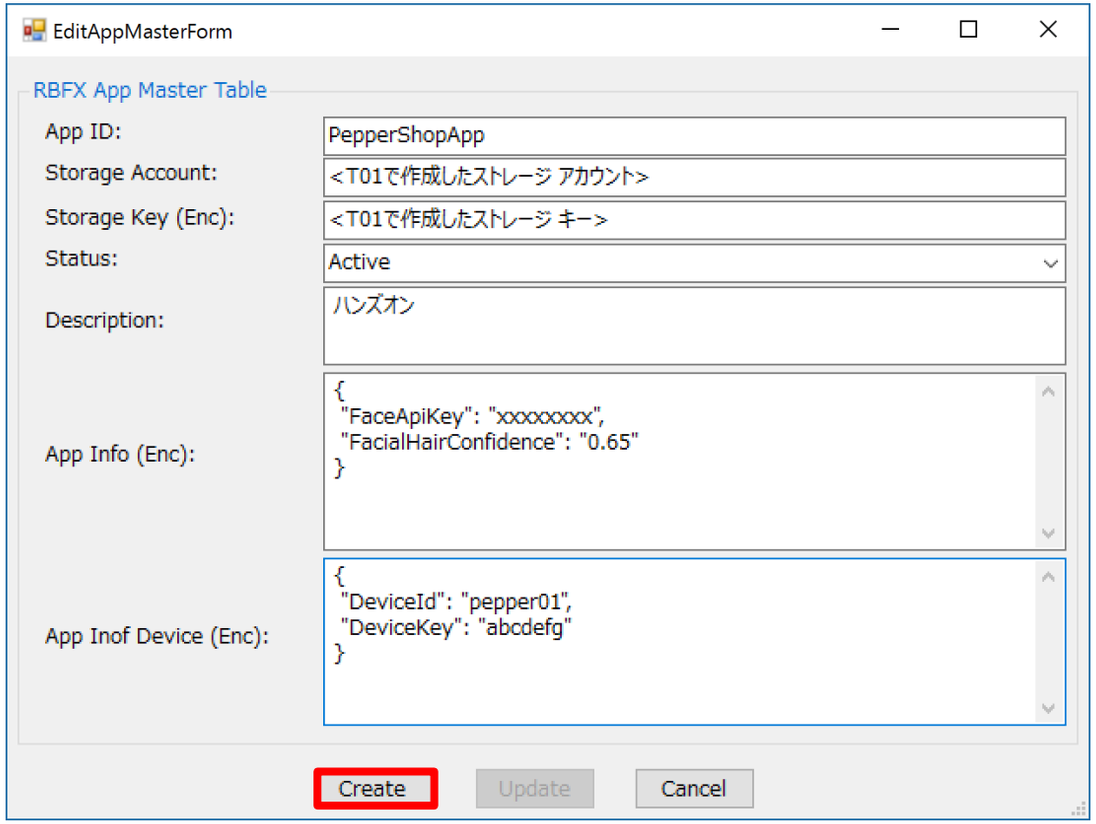
***2 : Once the DLL is read from the BLOB, it remains cached in the FX's current directory unless the Registered_DateTime of [RBFX]. [AppRouter] is changed**

T09_02 : App Master settings

Launch the Cloud Robotics Definition Tool (Management Tool) from the shortcut

- Open the App Master tab
- Set the encryption passphrase of Cloud Robotics FX set in T05_04 to [Encryption Passphrase] and [Save]
- Press the [Create] button, make the following entry in the App Master Edit form, and press the [Create] button

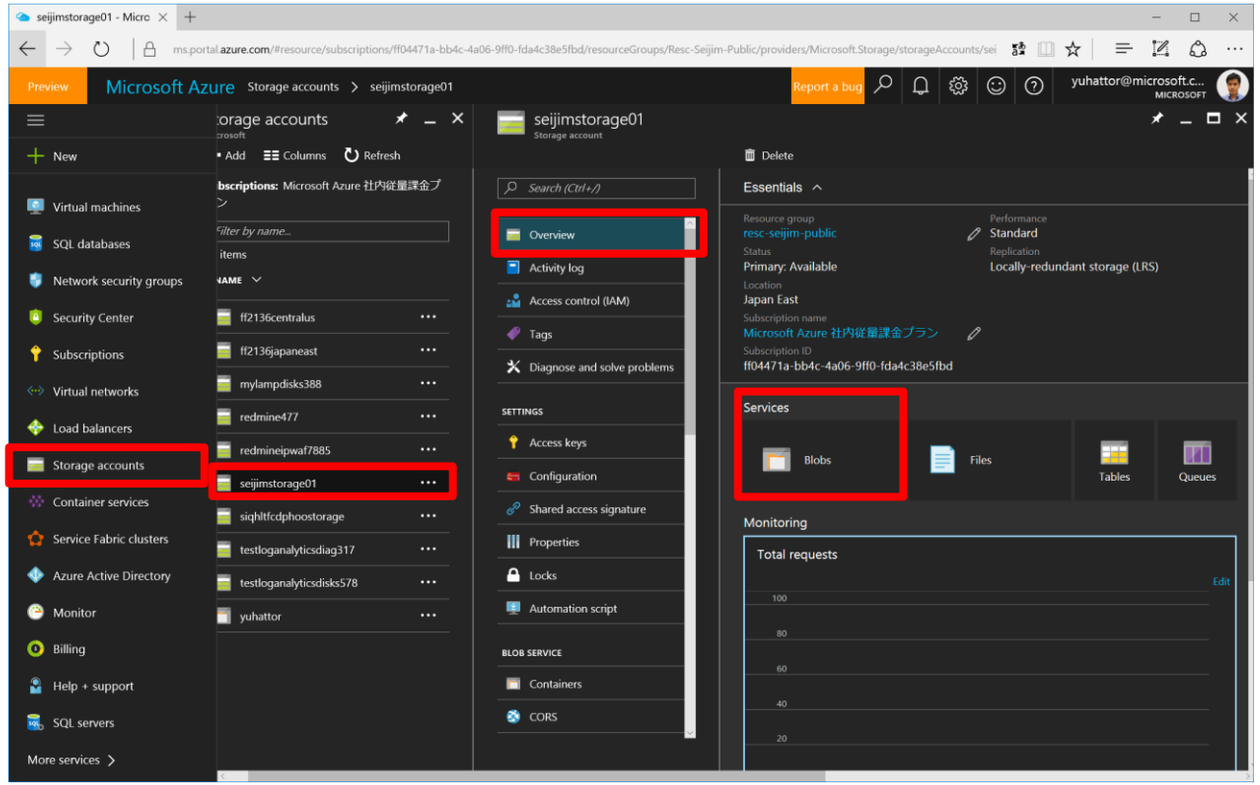


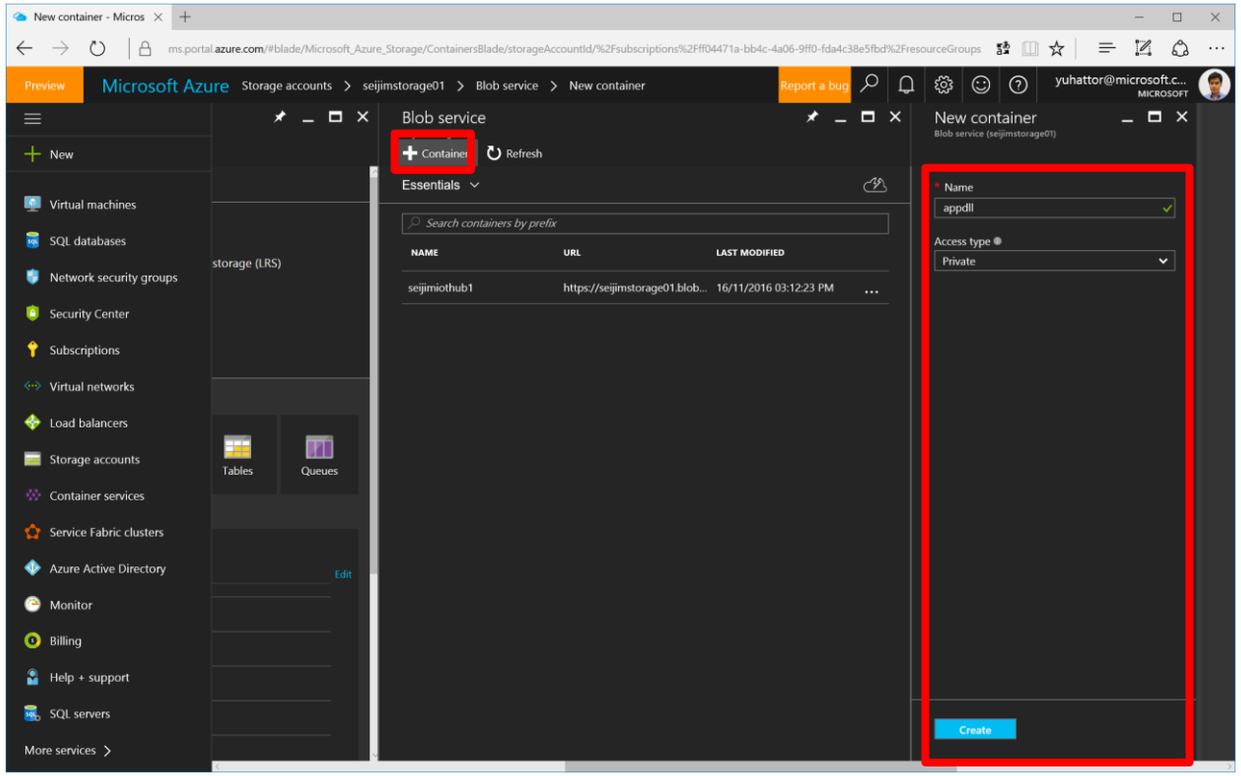


T09_03 : Create BLOB container for application (DLL)

Open the Azure portal

- Select [storage account] → [corresponding storage account] → [summary] → [BLOB]
- Click [+ Container], select Container Name = {appdll}, Type of access = {Private}, and click Create

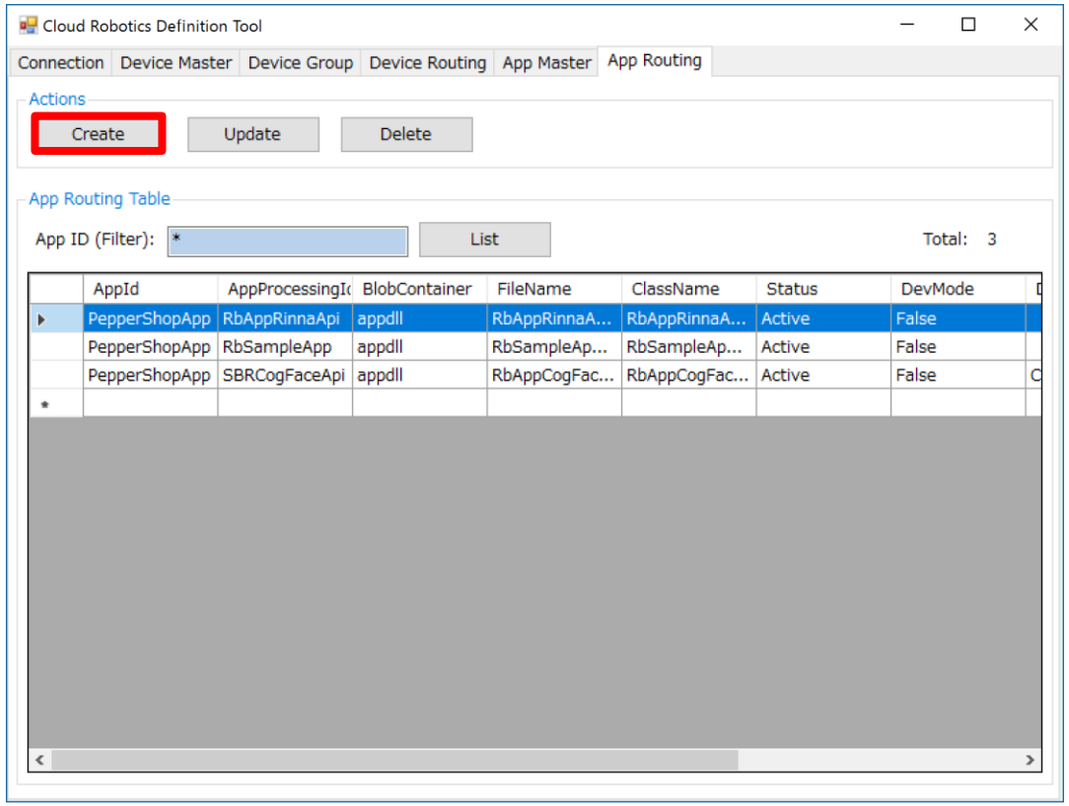


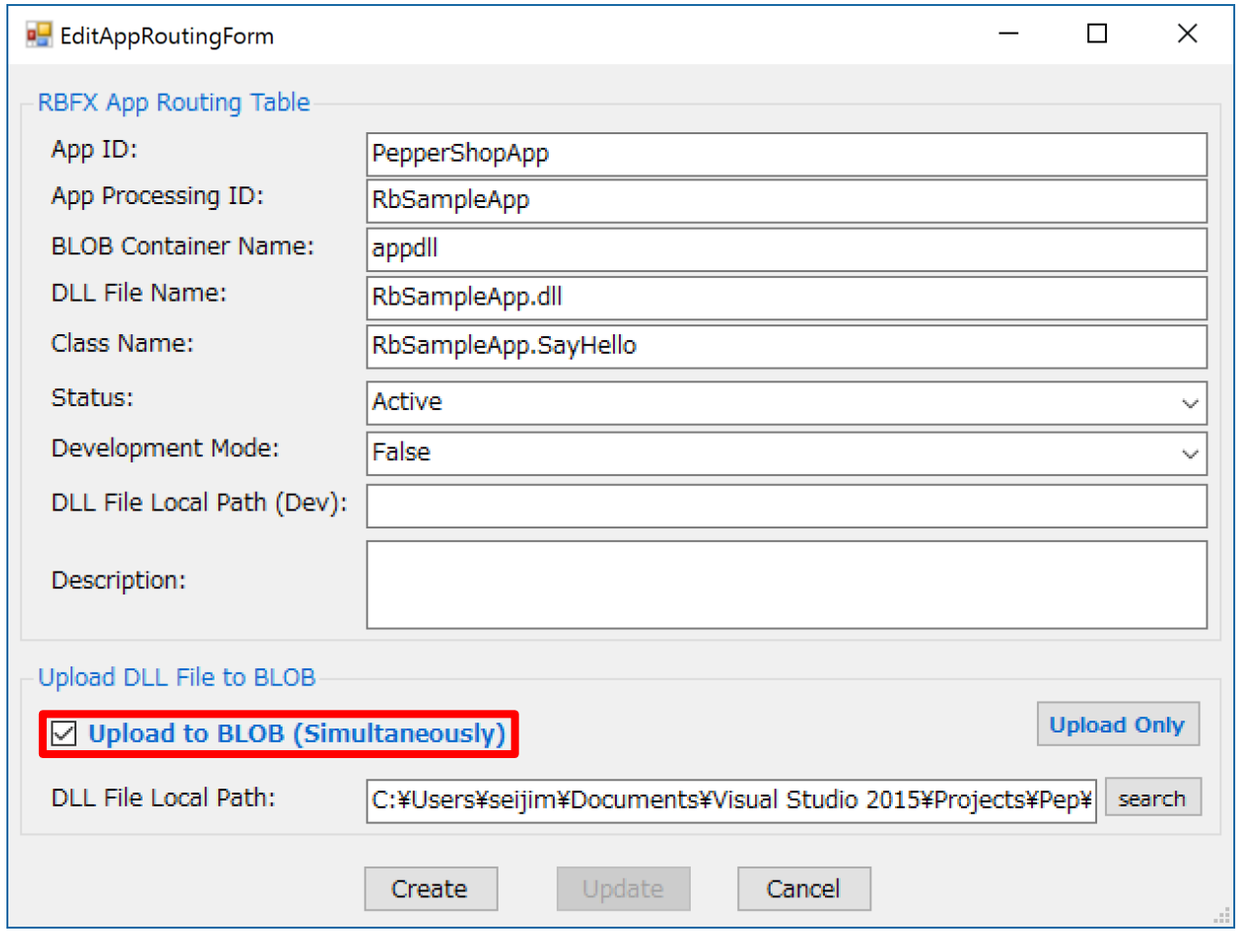


T09_04：App Routing settings and application deployment

Open the App Routing tab of Cloud Robotics Definition Tool

- On the App Routing tab, press the Create button
- In the App Routing edit form, make the following entries
- Select the RbSampleApp.dll file with [search], check [Upload to BLOB (Simultaneously)], and press the [Create] button

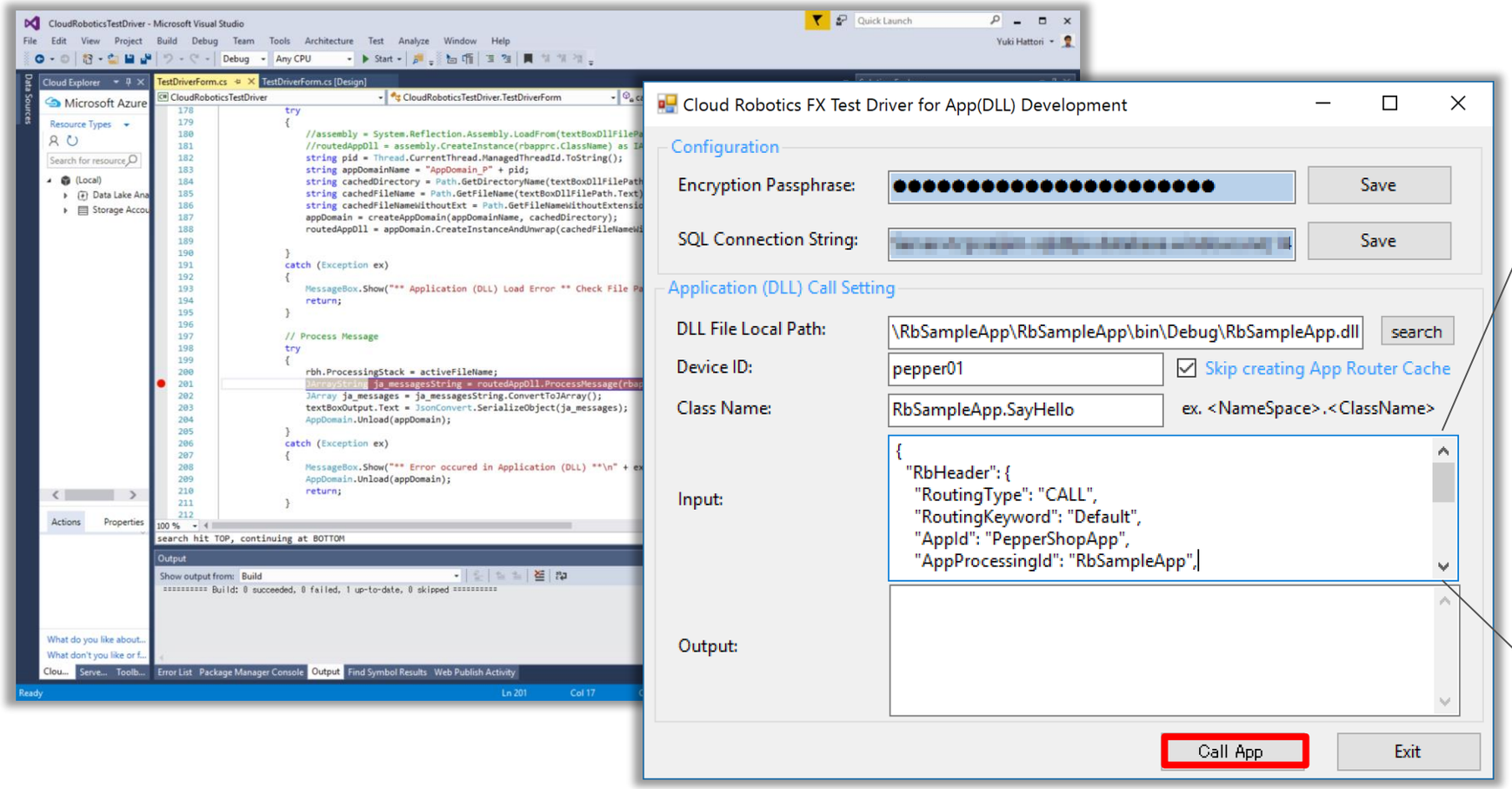




T09_05 : Unit test of application (DLL)

Open CloudRoboticsTestDriver solution file

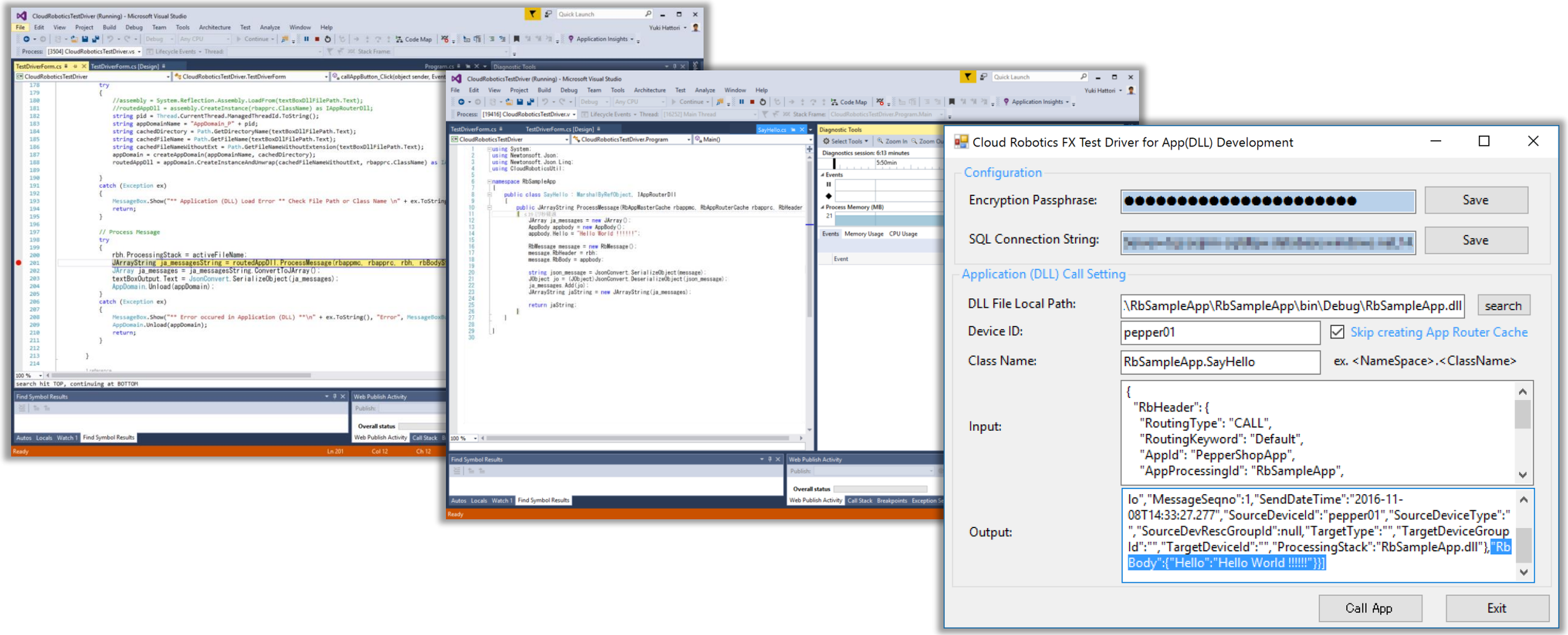
- Open "CRSDKv1¥CloudRoboticsTestDriver¥CloudRoboticsTestDriver.sln" and build it
- set a breakpoint in the "JArrayString ja_messagesString = routedAppDll.ProcessMessage (....)" Line
- Press [▶ start] button
- Set the encryption passphrase of Cloud Robotics FX set in T05_04 to [Encryption Passphrase] and [Save]
- Set the SQL connection string of Cloud Robotics FX set in T05_04 to [SQL Connection String] and [Save]
- Set [DLL File Local Path], [Device ID], [Class Name], [Input] and press the [Call App] button




```
{
  "RbHeader": {
    "RoutingType": "CALL",
    "RoutingKeyword": "Default",
    "AppId": "PepperShopApp",
    "AppProcessingId": "RbSampleApp",
    "MessageId": "SayHello",
    "MessageSeqno": "1",
    "SendDateTime": "2016-11-08 15:03:04.532"
  },
  "RbBody": {
    "visitor": "u002"
  }
}
```

T09_05 : Unit test of application (DLL)

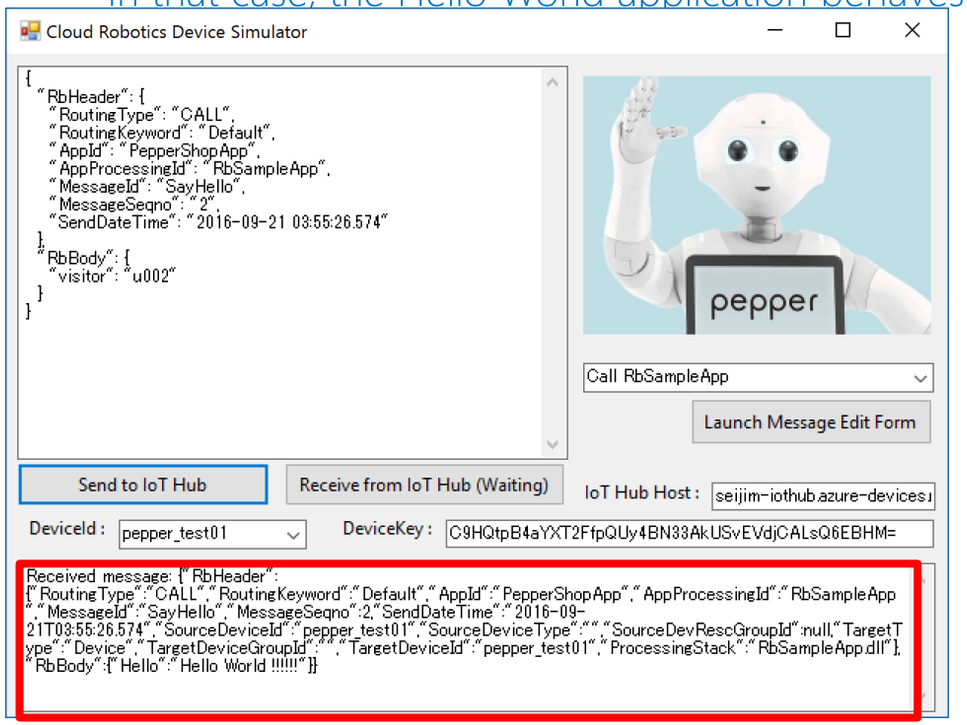
- When you are on the breakpoint, press [F11] and step into the Hello World app
- Debug the source code of Hello World application
- When it is executed to the end, the result is returned in the [Output] column
- If "Hello World !!!!!" is displayed, it is successfully completed.

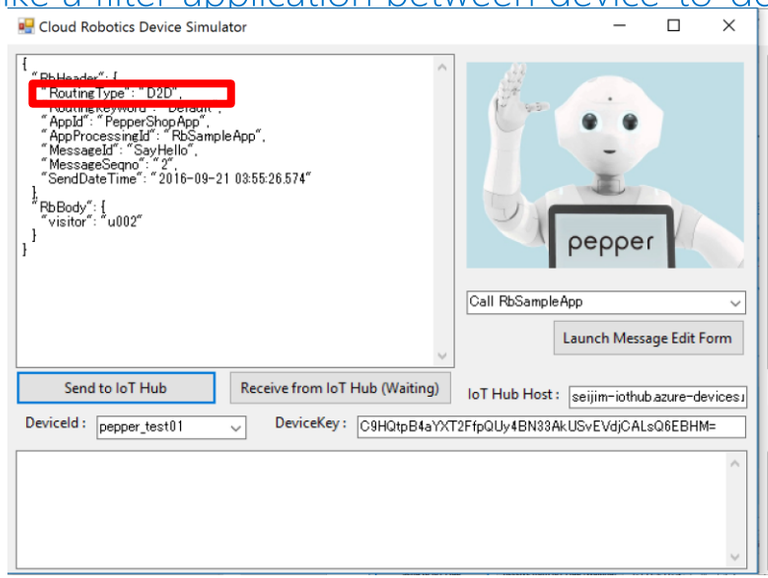


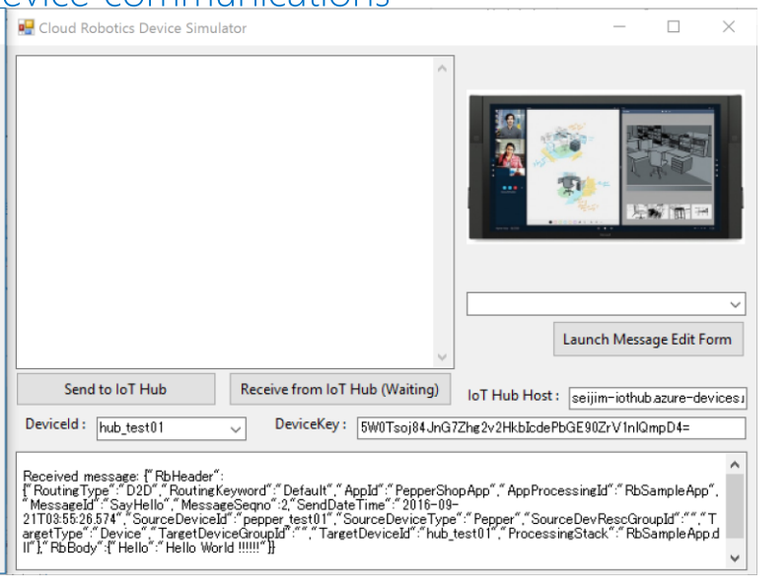
T09_06 : Testing deployed application (DLL)

Launch Device Simulator from Cloud Robotics Definition Tool

- Select "pepper_test01" on the "Device Master" tab and start Device Simulator
- Press the [Receive from IoT Hub (Waiting)] button
- Select "Call RbSampleApp" from the communication message template and press the [Send to IoT Hub] button
- If "Hello World" message is returned, it is successfully completed
- If you change the value of "RoutingType" of the communication message from "CALL" to "D2D", communication type will be device–to-device communication.
- In that case, the Hello World application behaves like a filter application between device-to-device communications







Tutorial 10

Stream Analytics settings

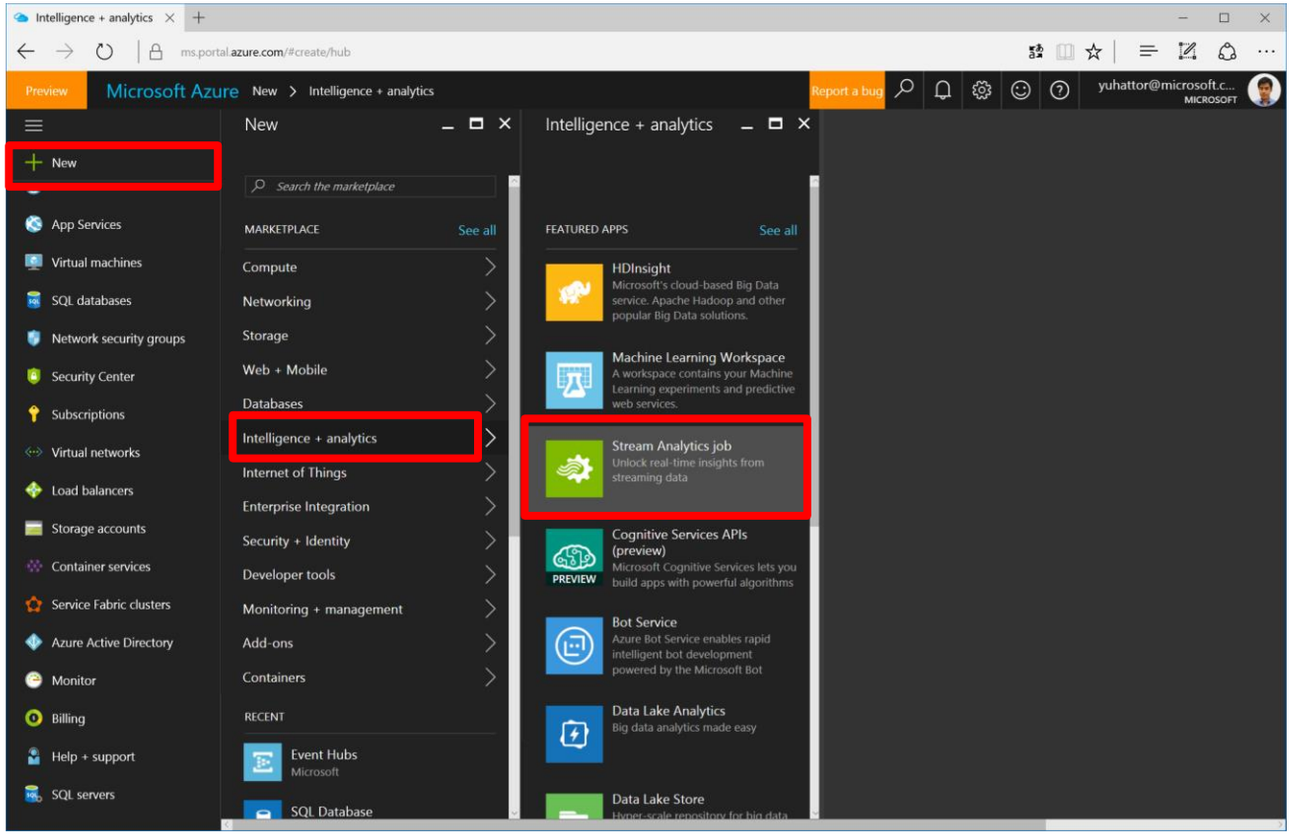
T10_01 : Create a Stream Analytics Job

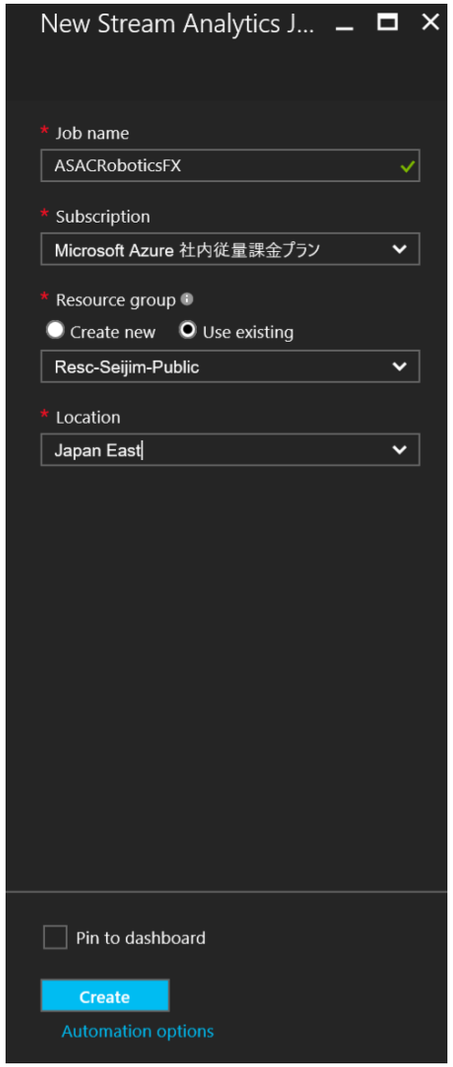
Open the Azure portal

- Select [+ New] → [Intelligence + Analytics] → [Stream Analytics Job]

Create a new Stream Analytics Job with the settings below

- [Job name] = {ASACRoboticsFX}
- [Resource group] = {Set value of T01_02}
- [Location] = {East Japan or West Japan}

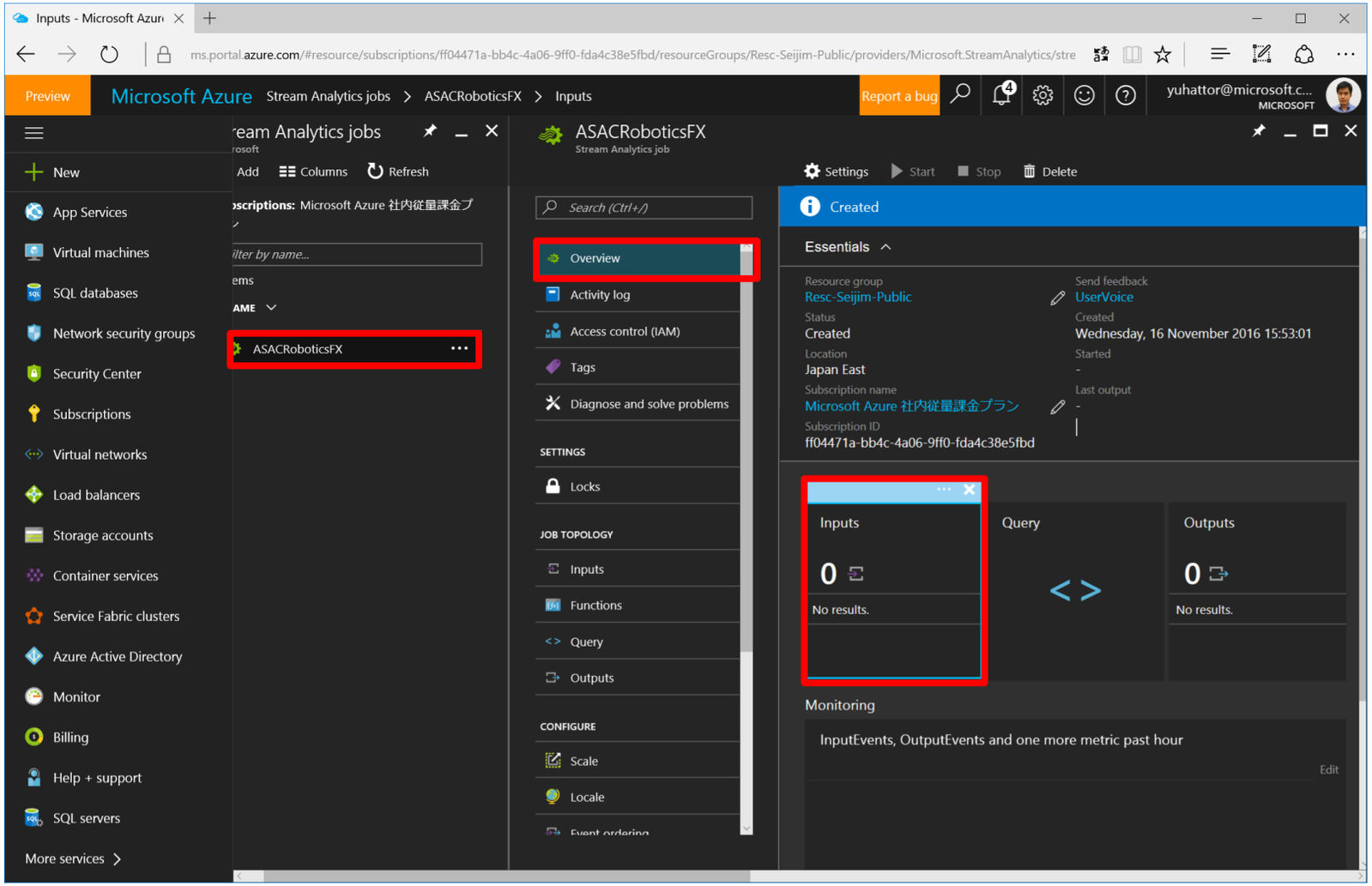




T10_02 : Input setting of Stream Analytics job

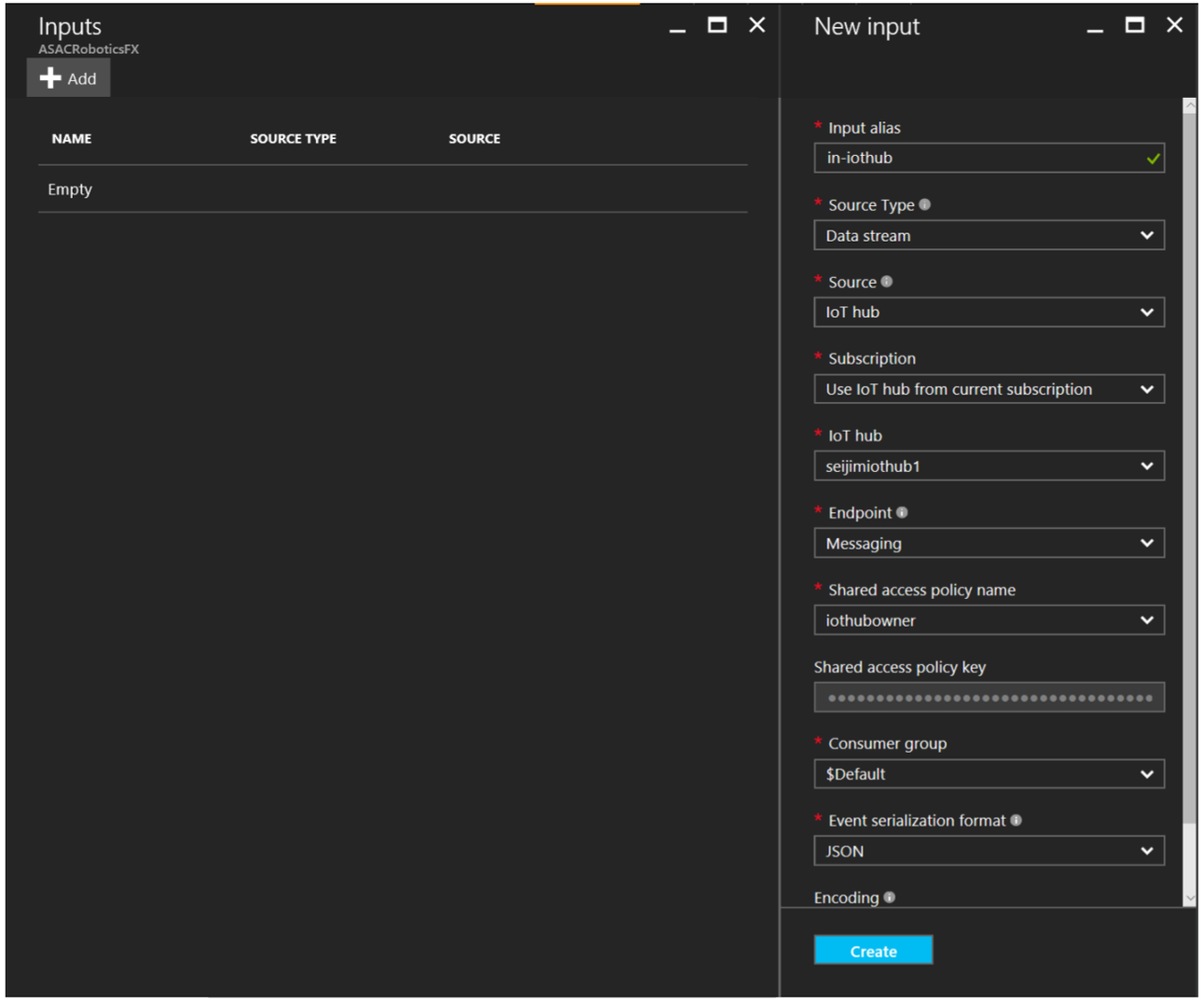
Open [Stream Analytics Job] from the Azure portal

- Select the relevant job and select [Overview] → [Input]



T10_03 ： Input setting of Stream Analytics job

Press [+ ADD], set parameters, and press the [Create] button
[Input Alias] = {in-iothub}
[Source type] = {Data stream}
[Source] = {IoT Hub}
After that, leave the default



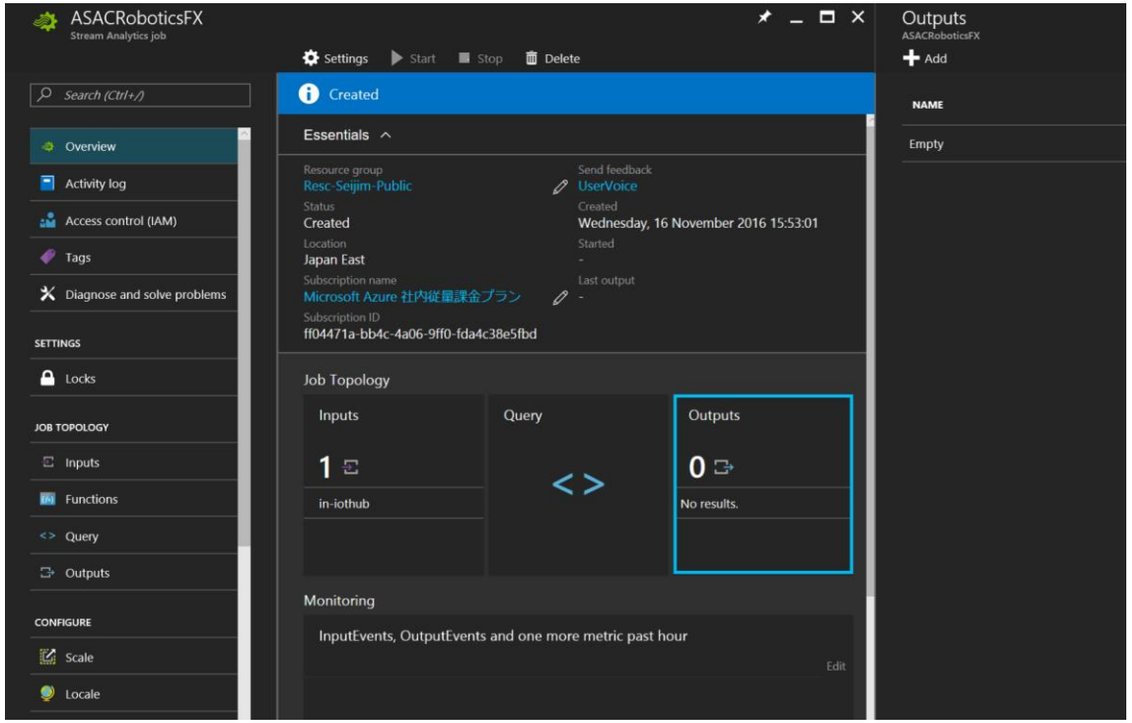
T10_04：Stream Analytics job output setting 1

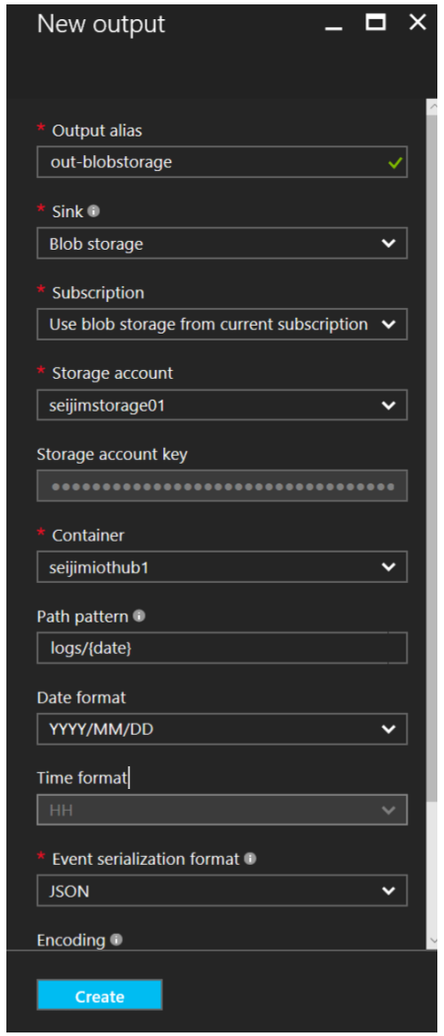
From the Azure portal, press [Stream Analytics Job] → [Summary] → [Output] → [Add]

- create a BLOB container "crobologs" in the storage account created in advance in T01

Set the output parameters to the BLOB and press the [Create] button

- [Output alias] = {out-blobstorage}
- [Sink] = {Blob storage}
- [Container] = {crobologs}
- [Path pattern] = {logs / {date}}

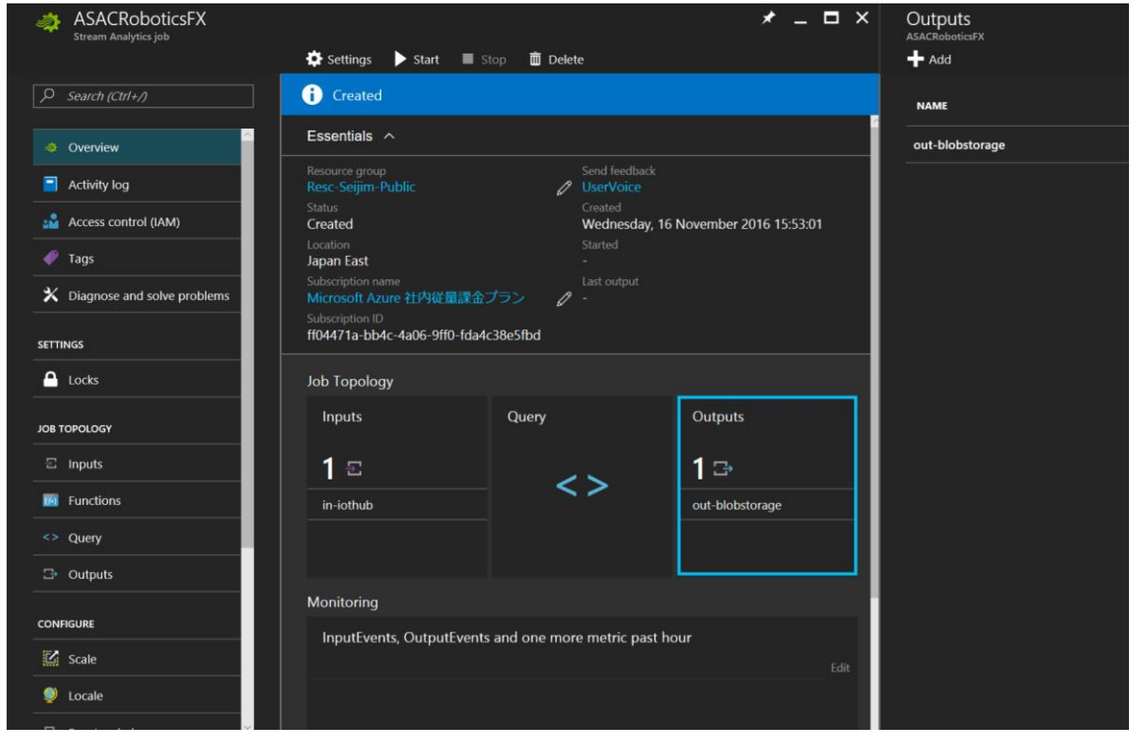


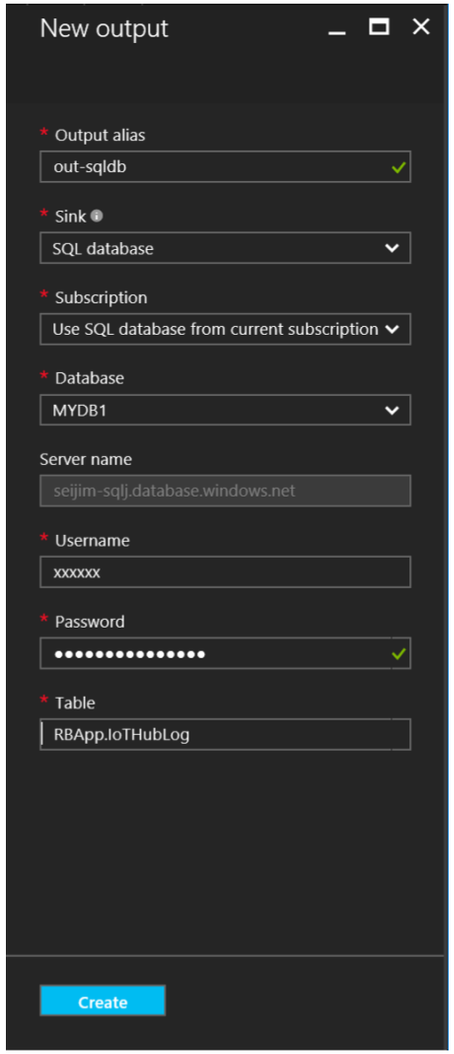


T10_05： Stream Analytics job output setting 2

From the Azure portal, press [Stream Analytics Job] → [Summary] → [Output] → [Add]
Set the output parameter to SQLDB and press the [Create] button

- [Output alias] = {out-sqldb}
- [Sink] = {SQL database}
- [Database], [User name], [Password] = {SQLDB created in T03}
- [Table] = {RBApp.IoTHubLog}





T10_06： Stream Analytics job output setting 3

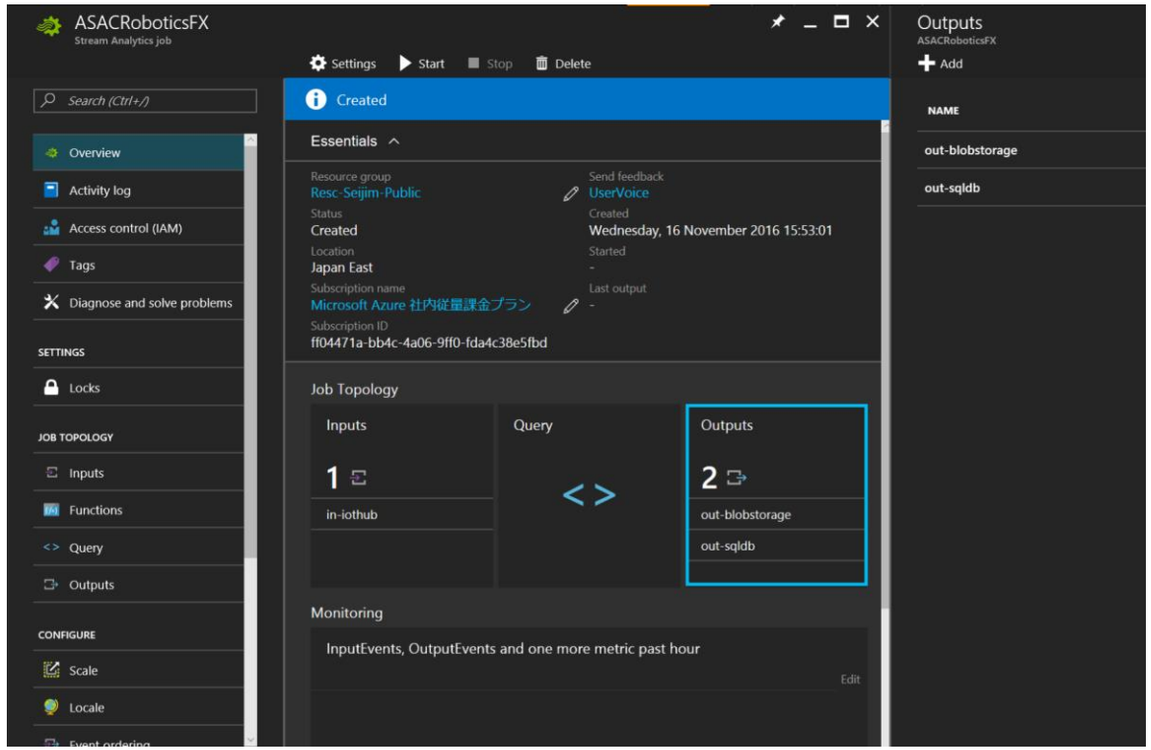
From the Azure portal, press [Stream Analytics Job] → [Summary] → [Output] → [Add]

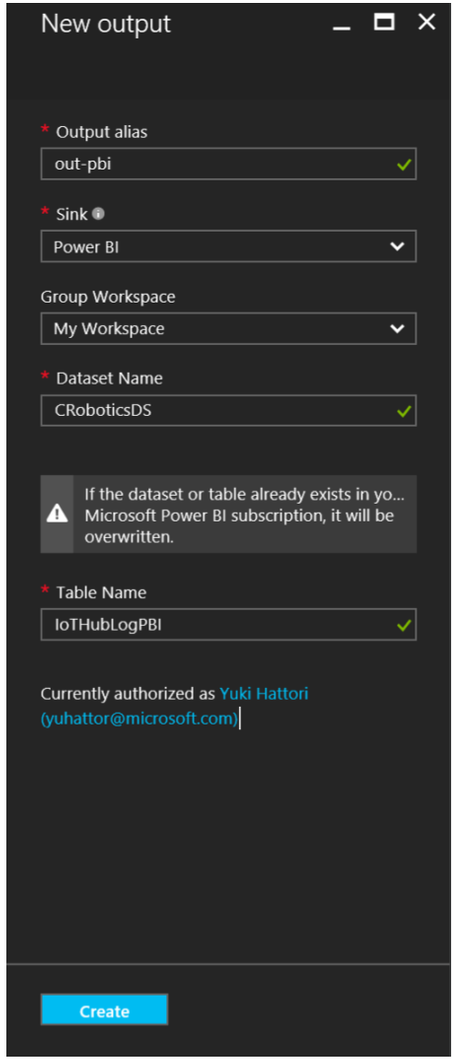
Set output parameters to Power BI

- [Output alias] = {out-pbi}
- [Sink] = {Power BI}

After implementing authentication to Power BI, set the following and press the [Create] button

- [Dataset name] = {CRoboticsDS}
- [Table name] = {IoTHubLogPBI}

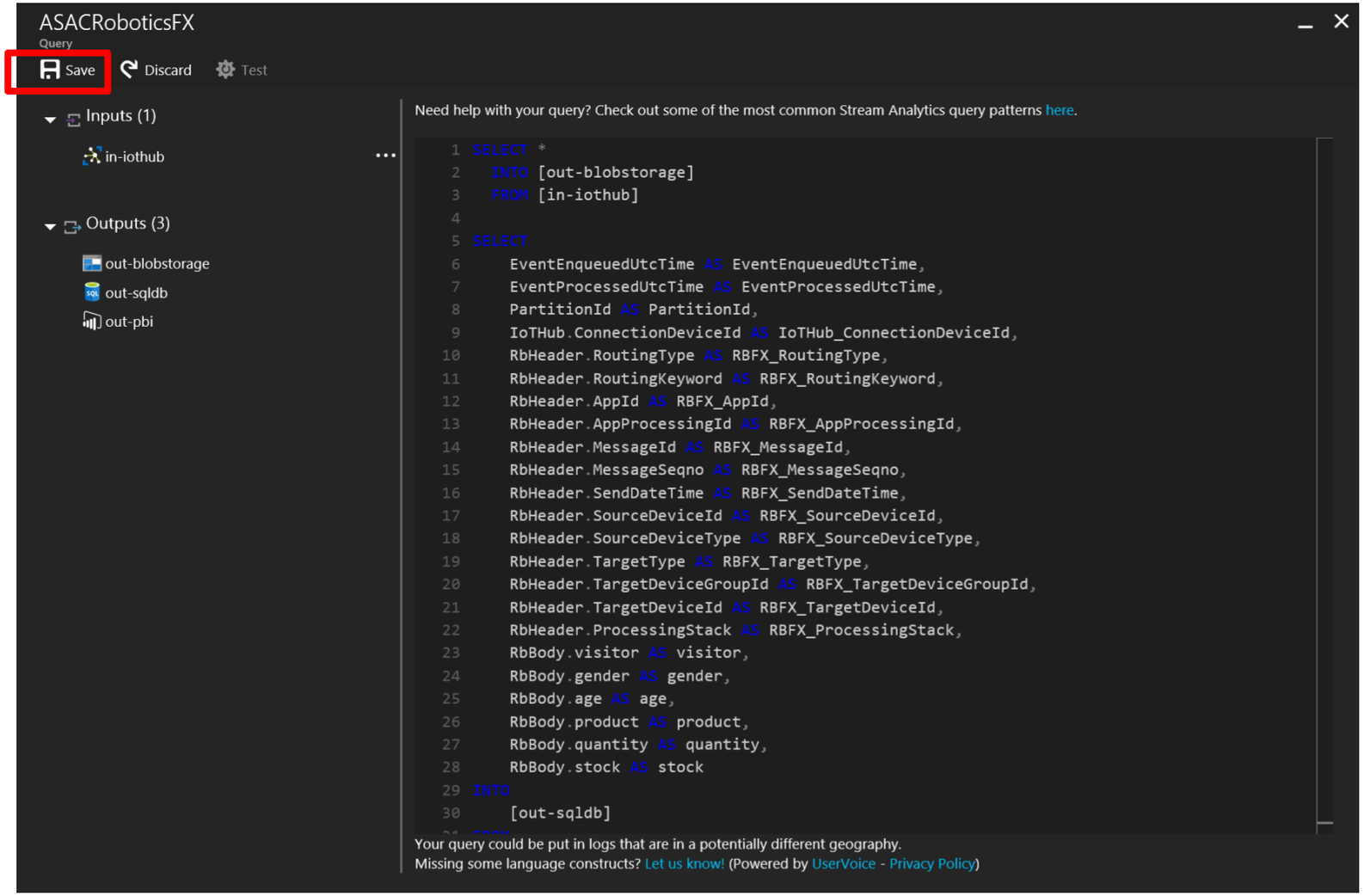




T10_07 : Query Settings of Stream Analytics Job

Press [Stream Analytics Job] → [Summary] → [Query]

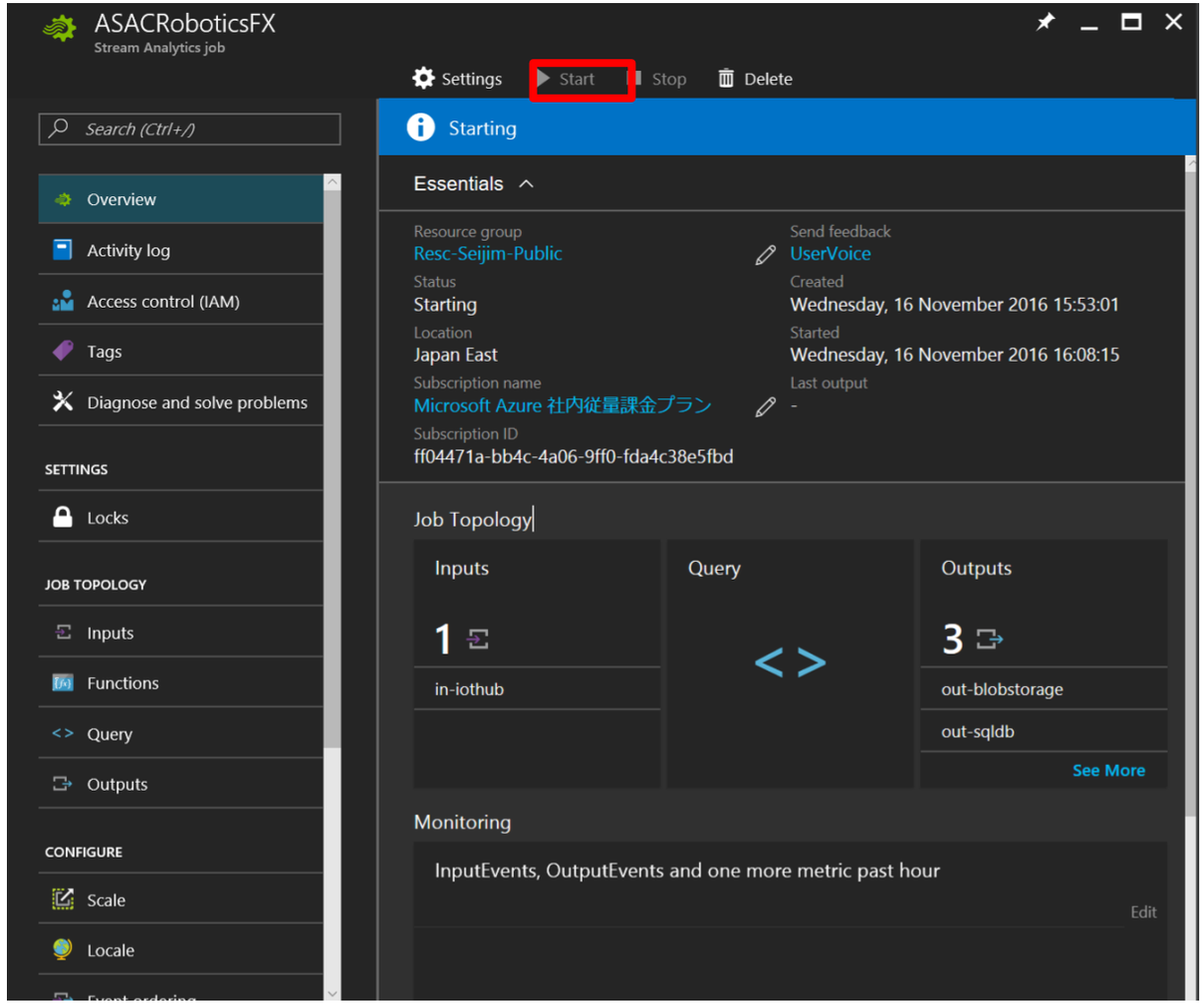
- Open "CRSDKv1 ¥ SQL_ASA_Definition ¥ 21_ASA_Query.txt" with a notepad etc., copy, paste and press [Save]



T10_08：Starting a Stream Analytics job

Press [Stream Analytics Job] → [Summary] → [Start]

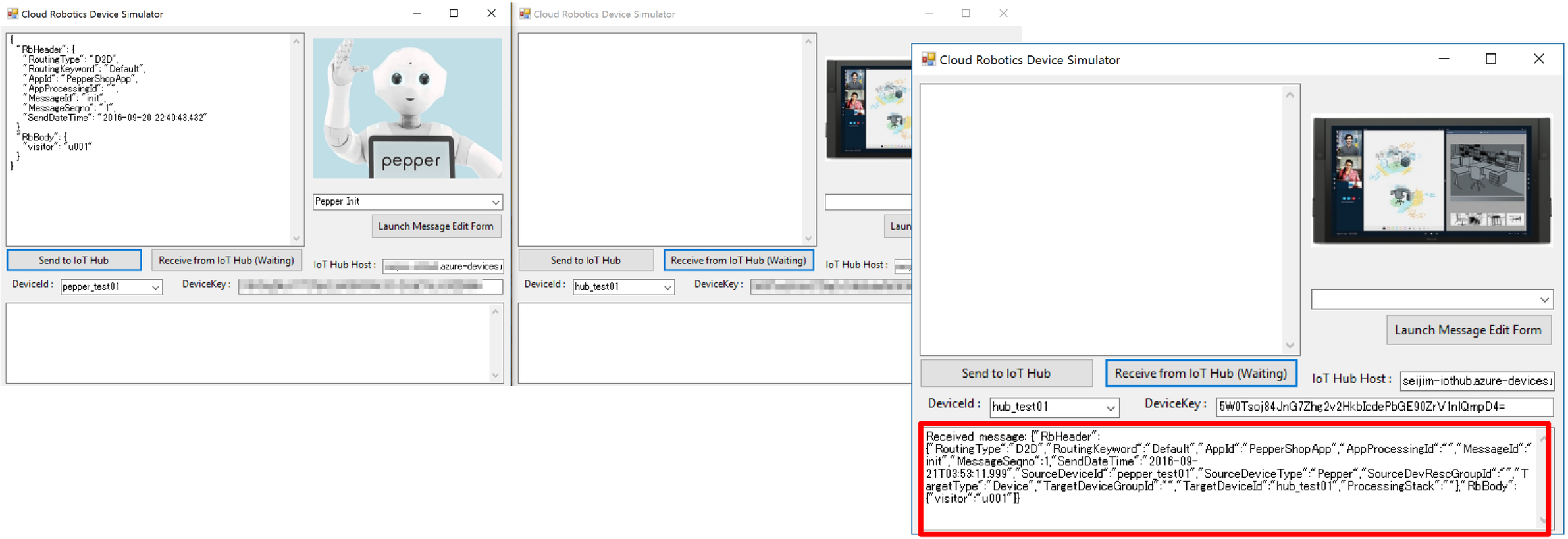
- The Read Point of IoT Hub will be asked, then select "start time" at the first startup and "last stop time" after the second time



T10_09 : Stream Analytics test

Perform device-to-device communication using Cloud Robotics Device Simulator

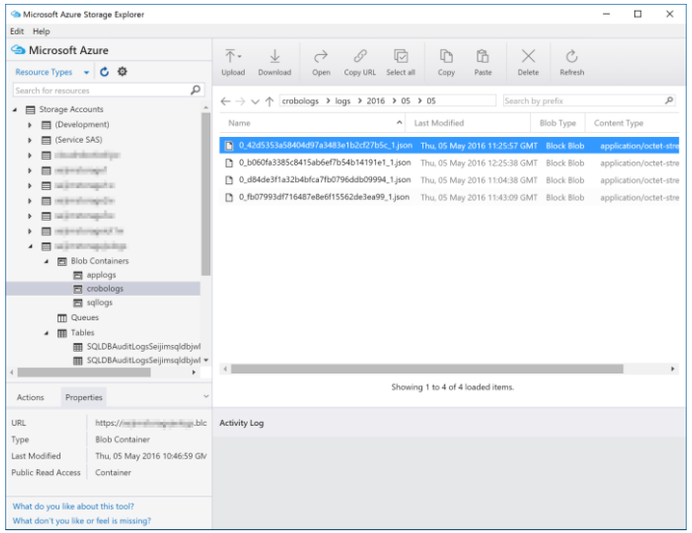
- Press the [Receive from IoT Hub (Wating)] button in the simulator of "hub_test01" (Surface Hub)
- Press the "Receive from IoT Hub (Wating)" button in the simulator of "pepper_test01" (Pepper)
- Send various messages from both simulators using the [Send to IoT Hub] button

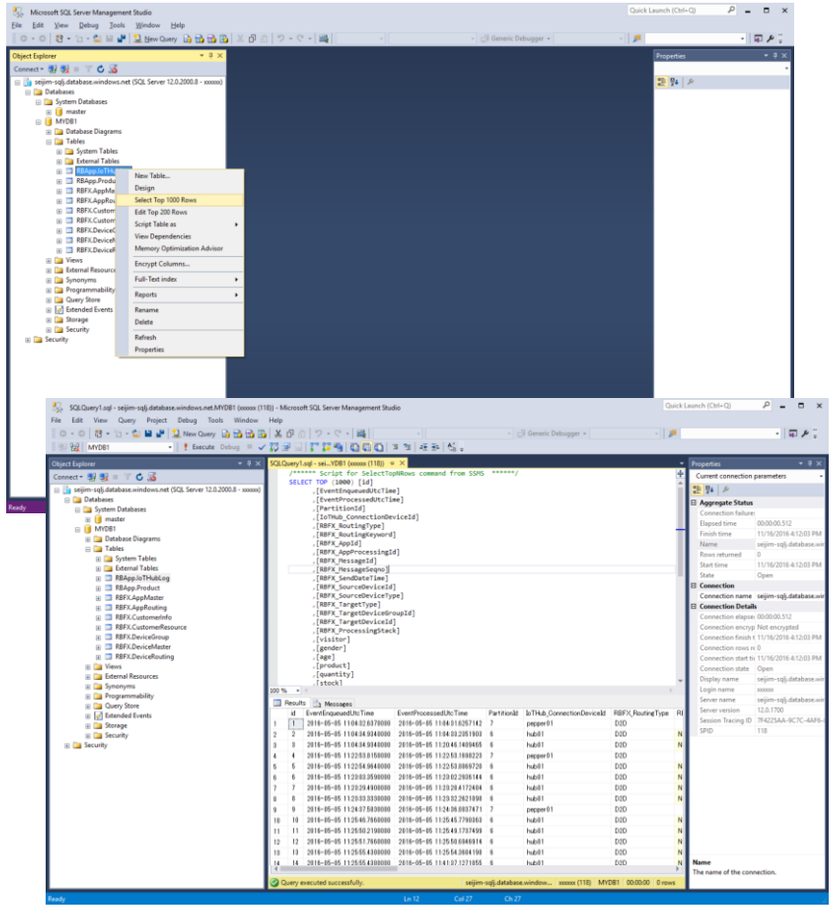


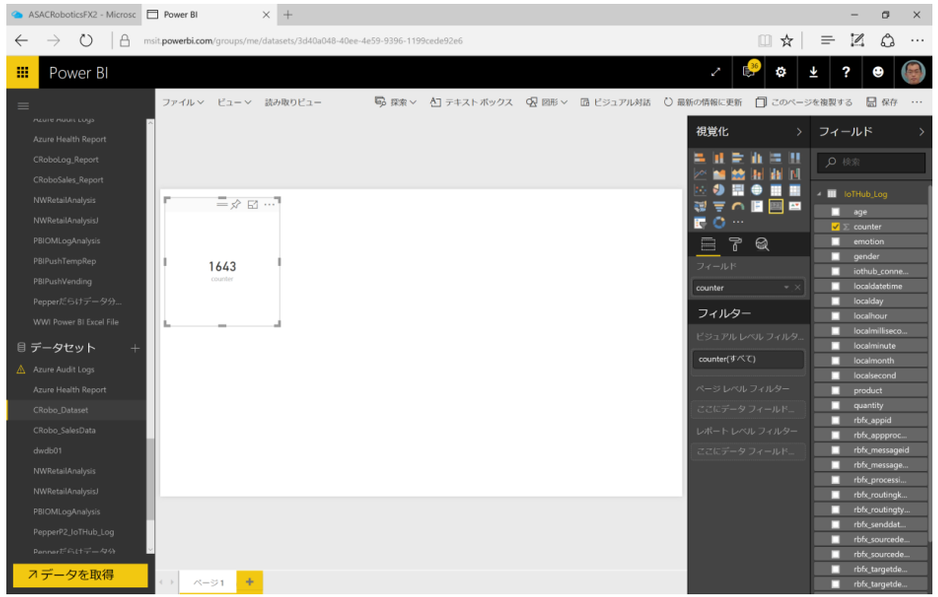
T10_10 : Confirmation of output result of Stream Analytics

Confirm the results for BLOB output, SQLDB output, Power BI output

- BLOB : In Microsoft Azure Storage Explorer, check the contents of the storage account and BLOB container to be outputted
- SQLDB : In SQL Server Management Studio, check the contents of the table to be output
- Power BI : Sign in to powerbi.com and check if the dataset is created



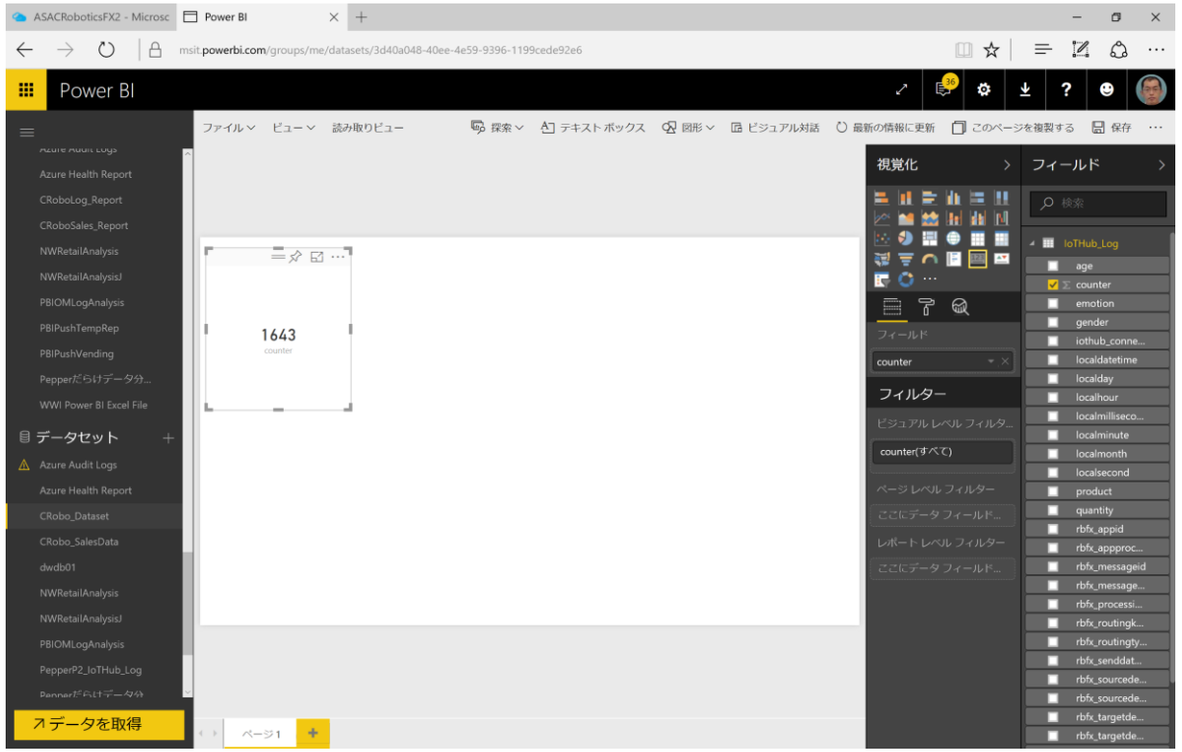


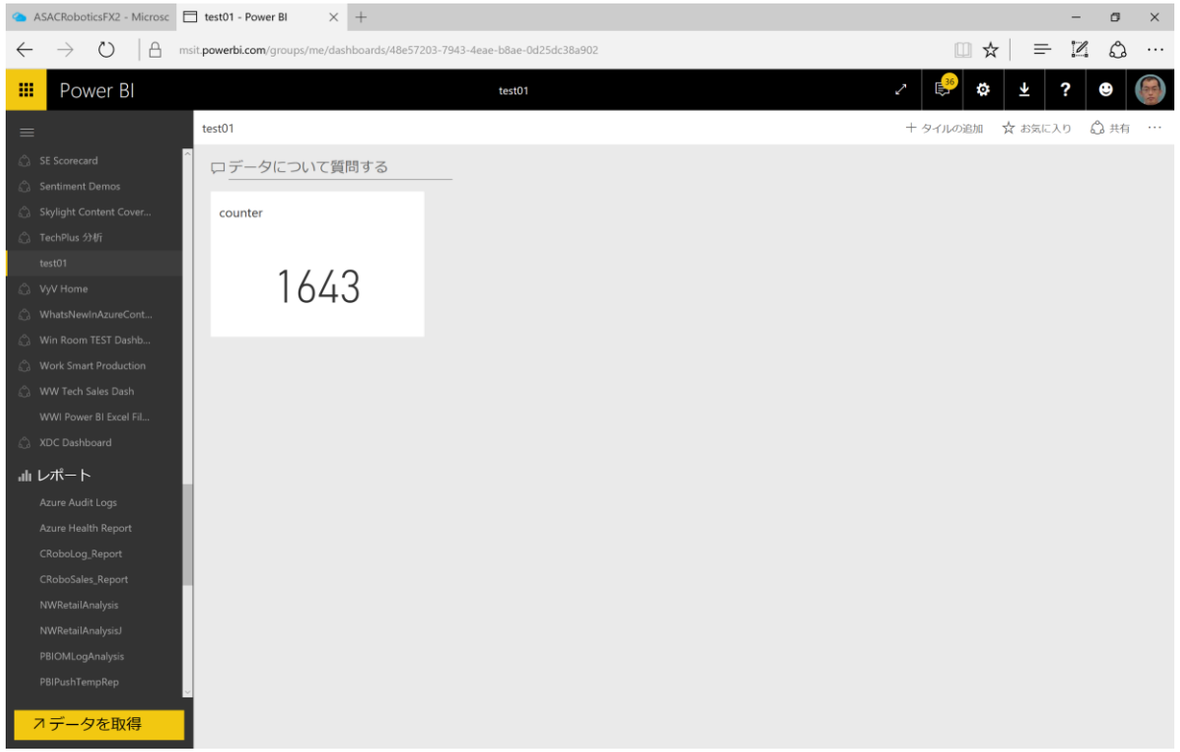


T10_11 : Power BI real-time dashboard

If you select the relevant data set, the report creation screen will be displayed

- Select "counter" from the field and make the visual representation "card"
- Name and save this report and press the "pin" button at the top of the card
- Select "New Dashboard" and give it a name
- When sending and receiving with the simulator, you can see that the dashboard counter value changes in real time





Microsoft

© 2016 Microsoft Corporation. All rights reserved.